COLOR BASIC UNRAVELLED II

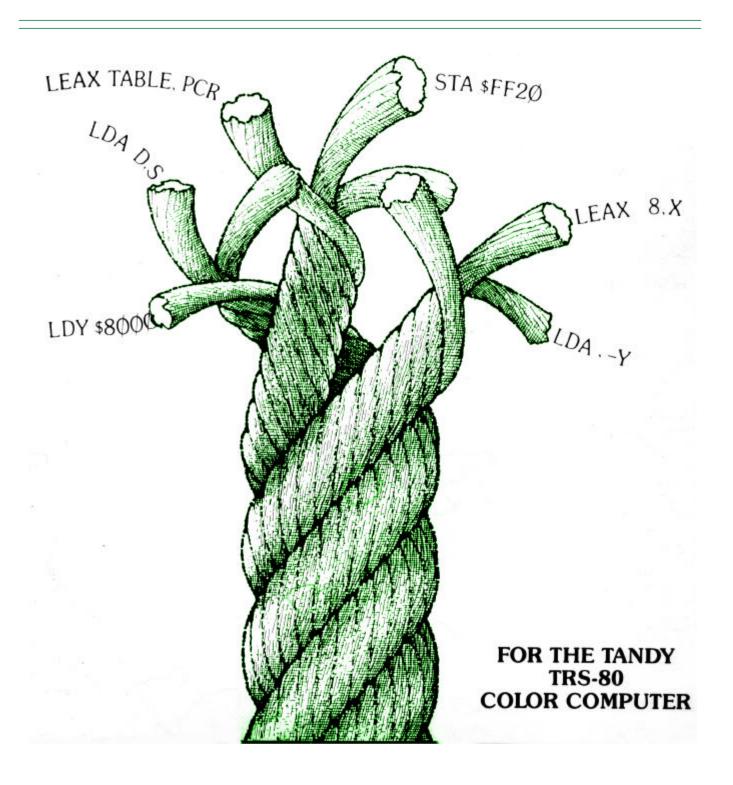


TABLE OF CONTENTS

1	FUKE	WORD								
2	INTRO	DDUCTION								
3	COLOF	R BASIC AN INTERPRETER								
4	INTER	INTERPRETER MECHANICS								
5	VARIA	ABLES								
6	CONS	OLE INPUT/OUTPUT								
		APPENDICES								
	Α	MEMORY MAP								
	В	DISASSEMBLY OF COLOR BASIC								
	С	BASIC ROUTINES AND ENTRY POINTS								
	D	FLOATING POINT ROUTINES								
	Е	BASIC S DATA/ASCII TABLES								
	F	MEMORY MAP DESCRIPTION								
	G	INTERRUPTS								
	Н	OPERATOR PRECEDENCE								
	Ι	BASIC 1.0 DIFFERENCES								
	J	BASIC 1.1 DIFFERENCES								
	K	ASCII CHART								

FOREWORD

Due to the many requests for the Unravelled Series produced by Spectral Associates, and the fact that these books are rare and no longer in production, I have taken it upon myself to reproduce them in electronic .PDF (Adobe Acrobat®) format.

I have re-disassembled the ROMs listed in this book, and added all the comments from the Original Extended Basic Unravelled Book. Some changes were made to make the book a little easier to read.

- The comments have been cleaned up some. In cases where a comments continued onto the next line, a * is placed in the Labels column, as well as a * at the beginning of each line of the comment. In cases where the previous comment used this format, a = was used. This was done in the original, but not all comments stuck to this format.
- 2. I have renumbered all the linenumbers. Each Appendix starts at Line 0001.
- 3. Some spell checking, and context checking was done to verify accuracy.
- 4. I used the Letter Gothic MT Bold Font. This allows for display of Slashed Zeros. I thought it important to be able to distinguish between 0 and 0.
- 5. All the Hex code now shows the Opcodes.

There were other minor changes that were made to make viewing a little better. If any discrepancies arise, please let me know so that I may correct the errors. I can be contacted at: mailto:wzydhek@internetcds.com

About Me

My name is Walter K. Zydhek. I've been a Computer Hobbyist since 1984 when I received my $1^{\rm st}$ Tandy Color Computer 2 for Christmas. It had 32K of ram, Cassette, and one Cartridge. I quickly learned to program in Basic and then moved into Assembly.

Over the next few years, I saved to purchase the Multi-Pak Interface, Disk Drives, Modem, OS-9, and various Odds and Ends.

I moved to Tampa Florida and in the move, My CoCo was damaged. I then replaced it with the CoCo 3. WOW what a difference. I added the 512K Ram Upgrade, A CM-8 color monitor, and joined a CoCo Club. Can anyone from Tampa, Florida tell me the name?

I had a couple of close friends that helped me explore the world of CoCo and by this time, I knew that my CoCo would be my friend forever. I give special thanks to Steve Cohn, who helped me get started with ADOS. Two other people who's names I can't remember were very beneficial to my mastering of the CoCo.

Shortly after getting my CoCo 3, I started BBS'ing.. Wow, a whole new world.. My knowledge just kept growing..

A few years later, I moved to Oregon, then to Phoenix, Arizona to attend school. I studied Electronics Technology at Phoenix Institute of Technology. In the second year, we studied Micro-processor Theory. For our labs, we just happen to use the Tandy Color Computer 3 (for studying 6809 Processors). I had it made. In this class I added an EPROM programmer/reader to my list of hardware. My favorite instructor, Gary Angle & I spent many hours sharing information on the CoCo. At one time, we shared a joint project to disassemble ROMs from an industrial machinery which used the 6809 Processor. Using the CoCo to read the ROMs to work with.

I even had a BBS running under OS-9 at one time. RiBBS I think it was. Very similar to QuickBBS and RemoteAccess BBS for the PC.

In 1991, I finally converted over to PC, but never forgetting my CoCo. About 5 years ago, My CoCo and all related material was stolen from me. And the CoCo world was just a memory.

In the last 2 Years, my love for the CoCo has re-kindled. I have been partially content to use a CoCo Emulator for my PC. I tried the CoCo 2 Emulator by Jeff Vavasour. This was OK, but a lot was left out. I then purchased the CoCo 3 Emulator. Much better, but would not use "Double Sided Disks". Although it did have a Virtual Hard Drive for use in OS-9.

I then wanted to 'better' the CoCo Emulator, add use of PC hardware, Add Double Sided Disk functionality, and even make it Windows Native, instead of a Dos Box. Unfortunately the I could not get the source code for the CoCo 3 Emulator.

I then turned to Paul Burgin's Dragon 2/Coco 2 Emulator. This had source code available and with a small \$20.00 donation, was able to get the source code to additional portions of his program. I have tinkered with it, but came to understand that I needed more info on the CoCo. I have looked all over the net and found quite a lot of useful information, but what I really needed was the Unravelled Series.

I was able to find someone that had Extended Basic Unravelled and Disk Basic Unravelled (He sent them to me for free). And a friend of mine had Super Extended Basic Unravelled (A copy I gave him years ago). Unfortunately, the books are not in the best of shape, and the type is hard to read, and with so many people looking for the books, I decided to re-do them in Electronic format.

I ask everyone that obtains copies of this electronic document to PLEASE give freely. These books are for educational/informational use only. These books are no longer in publication and Spectral Associates no longer in business. Do not use these books for financial gain, as that would most certainly abuse the Copyright Laws that I have already bruised by re-producing them.

Other than that, enjoy the books!! I'll add more information to them as I get it. I plan on adding more Memory Map information, as well as hardware info in the coming months.. But for now, take advantage of this fine resource.

Walter K. Zydhek

INTRODUCTION

BASIC Unravelled is a book that has specifically been written in order to provide the Color Computer user with a detailed, commented source listing of Color BASIC. Many entry points and useful routines for doing functions, which are required in machine language, have been detailed. Information contained in the book is extremely valuable and useful for anyone attempting to use BASIC integrated with machine language routines of their own. If there are some functions or facets of BASIC, which are too slow or awkward for the user, he needs to have a good idea as to what is going on in Color BASIC in order to be able to merge his routine with the routines, which are in color BASIC. There are many times when a person wants to know exactly what is going on in BASIC for a certain function such as clearing the screen or outputting a character to the screen. The information provided in BASIC Unravelled will allow the user to determine exactly what BASIC is doing under these circumstances. He will get an extremely good, in depth, basic knowledge of BASIC and be able to use that in any application he has in mind.

This book will not explain how to make the BASIC interpreter or give a detailed in depth knowledge of how a BASIC interpreter works. It assumes that the user is an experienced machine language programmer, understands 6809 assembly language inside and out, and will understand the nuances and programming terminology which is used in the comments included with each BASIC program line. Do not attempt to use BASIC Unravelled as a textbook in order to teach yourself how to write a BASIC interpreter. If you are a very good programmer you will be able to write your own BASIC interpreter following the in depth reading of the assembly listings and the comments included in this book, but it should not be taken as a text on how to write a BASIC interpreter. The book is primarily designed to explain Color BASIC so that somebody who has a fair knowledge of how an interpreter works will be able to determine exactly how Color BASIC works. BASIC Unravelled will explain major operating formats of the most useful routines in BASIC and will identify the tricks, which Microsoft has used in programming Color BASIC. If the reader has any questions concerning the hardware of the Color computer, he's referred to the FACTS book, published by Spectral Associates. This book contains detailed descriptions of the hardware of the Color Computer and how one uses software in order to enable or disable the various hardware functions of the computer.

BASIC Unravelled will deal specifically with Color BASIC version 1.2 which is the version of Color BASIC released by Radio Shock, as of October 1983. The two earlier versions, version 1.0 and version 1.1, have only minor differences in relation to version 1.2. These differences are described in detail in the appendices, and if the reader has any questions in regard to version 1.1 and version 1.0 is referred to those appendices. Extended BASIC and Disk BASIC are covered in the two final books of the BASIC Unravelled sequence published by Spectral Associates. Any questions that regard explicitly to Extended BASIC and Disk BASIC will be covered in those books.

COLOR BASIC - An Interpreter

Color BASIC is a computer program, which is written in machine language, is very complex, and is extremely difficult to understand without some kind of helpful information. The idea behind writing a program, such as BASIC, is that BASIC is very easy to understand for the beginning user. Machine language, unfortunately, is very difficult to use and takes considerable amount of practice in order to get familiar with it. Therefore, BASIC is the language, which is provided with most computers when they are sold to the general public. As the user gets more and more familiar with BASIC, more and more questions generally arise as to how BASIC functions. That is one of the main purposes of the book -- to explain to the user exactly how Color BASIC, the Interpreter, works.

It is assumed that the reader is familiar with the manner in which the Interpreter functions. He at least knows the basic overall method of how an interpreter works in that the lines must be numbered, the interpreter executes these lines one after the other, and transfers control with GOTO, GOSUB and other similar statements. BASIC is an interpretive language related to the direct commands we are executing. BASIC executes a command by taking the last line typed to it and analyzing the line working from left to right looking for keywords and expressions, which it recognizes. Every time it encounters a keyword such as PRINT (or ? which is the abbreviation for PRINT), it interprets this word into a command, which means something to BASIC. Command words are stored in memory with bit 8 set to tell BASIC that it is a command word, or keyword (token). As a program line is entered into RAM memory through the use of the enter key, BASIC takes the line number and searches through memory, until it finds the same number, or the number just greater. If it is the same line number, then the entire line in memory is deleted and a new line is inserted into memory. In the preinterpreted state all the keywords are replaced with the single character token of the keyword. This allows the interpreter to store commands in the most memory efficient form. The only data stored is the data typed in by the programmer such as strings, pointers to the variables, and the keywords. PRINT, even though it takes five characters to type, only takes one character in memory.

BASIC is called an interpreter because the actual execution of the instructions is done by analyzing the keyword that needs to be executed in the program line, then executing that keyword under the control of a series of subroutines. This is a trade-off, which results in very memory-efficient storage programs but longer execution times that would be true of a machine language program. Because Color BASIC uses tokens in memory and stores them on I/O devices whenever a program is loaded and saved, the actual coding of data on tape or in memory is not transferable to other Machines. It is generally not possible to use BASIC instructions typed in from other machines. It is not assumed that the reader is very familiar with all the weaknesses and strengths of the BASIC interpreter as opposed to a compiled language. No effort will be made to explain the differences between compilation and interpretation except to make note of the fact that many of the weaknesses of the BASIC interpreter stem from the fact that it is not compiled; that is, that the program is not converted into machine language and executed in one pass after it is converted into machine language. Each time a statement has to be interpreted with the BASIC interpreter, the interpreter must look up the functions that need to be interpreted, find out what they are, calculate any numerical results that are necessary as a result of the interpretation, print things to the screen and so forth, and then continue to the next statement. This is one of the main weaknesses of an interpreter-it is slow. Every time a statement has to be interpreted the some slow process has to take place. A perfect example of this is the determination of the value of a variable. BASIC stores its variables in a large table directly after the BASIC program. These tables have the variable tagged by its name, that is the one or two character ASCII sequence which is defined in the program such as: AA, Al, X, Y, etc. These variables are listed one after the other in the variable table and every time the BASIC program makes reference to a variable, BASIC must start at the beginning of the variable table and search through the entire length of table until it determines where that variable is. If a program were compiled, the program would know exactly where the variable is and wouldn't have to go searching through the table in order to find it. Obviously if the variable was located near the end of a very long variable table, a substantial amount of time will be consumed every time the BASIC program makes reference to this variable. This is one reason why it's convenient to put the variables which are the most often used in a BASIC program, at the very beginning of the program. Another example of the slowness of an interpreter is every time you make program control transfers such as GOTO or GOSUB, the program has to search through the entire length of the BASIC program in order to find where the destination line number is. If the line number happens to be just before the point where the BASIC program starts searching, the entire program will have to be searched through in order to determine where the program line is and then transfer control there. As you can see, this will waste a lot of time.

Why then, you would ask, do we use BASIC programs in the first place? The primary reason is because of the fact that BASIC is user friendly. It is simple to learn and it is simple to program. And even if it's slow, it still provides a very powerful tool for the user. It is very easy to develop and test programs and takes only a fraction of the time involved in what it would take to develop a comparable length machine language program. The penalty, of course, is the speed and the size of the final program.

INTERPRETER MECHANICS

The Interpreter has various statements, commands, and functions, which are used in order to process, manipulate or otherwise use data. The overall goal of any computer program is the manipulation and movement of data in the memory of the computer and the transference of that data to an input/output device such as the screen, a disk file, or a cassette tape. Commands will tell the Color Computer to do something with the program. Sample commands would be LIST, RUN, STOP, and CONTINUE. Statements are used to operate on the data or program, which is in the Color Computer at the time. Examples of statements are CLEAR, DATA, DIM, END, GOSUB, INPUT, and so forth. Functions provide another manner that BASIC statements can be used to control the Color Computer. Intrinsic functions provided by BASIC are used to operate on string or numeric data and produce a result, which will be useful. Many of these functions are mathematical functions or string manipulation functions which will form substrings based upon certain subsets of the string in question. The arguments of these functions are always enclosed in parentheses. The argument of any function is the value which is being manipulated by the function and sometimes there may be more than one argument in any particular function, such as MID\$, which may have three arguments. Often arguments may be left out and BASIC will supply default values. These default values can be found by looking at the routine in BASIC which controls that particular function and you can see whether or not a default value is allowed for. Sometimes a default value is not allowed, and if a value is not given BASIC will generate an error. Some examples of functions are ABS, ASC, SIN, COS, RIGHT\$, LEFT\$, etc.

The commands and functions of BASIC must be identified to the computer in a way that they can be understandable by the machine. The machine cannot understand PRINT, LIST, and RUN, it can only understand numbers. Therefore, there must be some way of identifying the commands and functions to the computer so that it knows that these are commands and functions. The method that most BASIC interpreters use in order to identify commands and functions is to identify them with a number from 128 to 255. The user will notice that these numbers are the equivalent of an 8-bit byte with bit 7 set. This is a very convenient way of identifying and abbreviating the commands and functions, because the numbers from 0 to 127 are the normal ASCII symbols used by BASIC. The numbers from 128 to 255 represent graphic symbols, which are rarely used in ASCII strings. These abbreviations for the BASIC commands and functions are called tokens. You will find in the BASIC listing a table of tokens and the respective addresses where command of BASIC is transferred when a certain token is encountered in an interpreted line. These are referred to as the dictionary of commands and the dictionary of dispatch jump addresses. Color BASIC created a problem when BASIC was written because of the fact that so many commands were required because of the graphics routines and the disk routines that 128 different commands would not suffice. Therefore, a novel method of expanding the number of tokens available by using the token \$FF as a special pre-token marker. There are two sets of tokens in Color BASIC, primary and secondary tokens. Primary tokens will have a value from 128 (\$80) to 254 (\$FE). If an \$FF token is encountered in an input line it signifies that the byte immediately following the \$FF is the secondary token in question. This can be confusing if you are not careful so you must be aware of the fact that secondary tokens require two bytes, an \$FF to identify it as a secondary token and then a number from 128 to 254 immediately after it which identifies the actual token number. Secondary tokens are used to keep track of the intrinsic functions, whereas the primary tokens are used to keep track of commands and statements.

When you type LIST, the computer lists your program; you see the words PRINT, LIST, NEW, LEFT, etc., spelled out on the screen for you. When the program statements are stored in the computer's memory these words are not spelled out. The

tokenized value of these words is what is stored in the memory of the computer. There are two routines in BASIC called crunch and uncrunch which will tokenize or detokenize the BASIC input line. When you list the line to the screen, uncrunch is called and the tokens are detokenized and converted into ASCII strings which are representations of the tokens. When you type a program line into the computer from the keyboard and then hit ENTER to store it into the computer's memory, crunch is called and it tokenizes the line. PRINT, LIST, etc., are crunched down from ASCII representations of those words into tokens. This explains why when you use a monitor to look at an actual BASIC program which is stored in the memory of the computer you will see ASCII strings and PRINT literals or the prompts for input statements spelled out as ASCII words, but you will see graphics blocks interlaced throughout your program. These graphics blocks are the tokens, which have been crunched by BASIC and stored in the computer. Later on you will see in the disk or in the cassette versions of input/output routines either crunched methods of saving the program or ASCII saves. The crunched method of saving the program is the normal method where the BASIC program is merely taken directly from the memory of the computer and stored onto the tape or disk. The ASCII save is where the program is taken from the memory of the computer, uncrunched and then saved on the tape. Generally, when you want to transfer BASIC programs from different computers, i.e., Radio Shack to an Apple, Atari or vice versa, you will have to move the files back and forth in ASCII format because all of the tokens for the different machines will have different values, not to mention different functions.

BASIC program lines are stored in RAM according to the following rules:

- 1. Start of text is a zero byte.
- 2. Each program line is preceded by a two-byte RAM link containing the address of the first byte of the next line and a two-byte line number.
- 3. The compressed (crunched) text is followed by a zero byte.
- 4. End of text is stored as two zero link bytes.

Now its time to investigate the process, which is used by BASIC in order to interpret the program line. There is a main program loop in BASIC, which is referred to as the command interpretation loop. In this loop commands and statements are evaluated. The token is decoded to determine where the routine is which must be jumped to in order to perform the particular function that needs to be evaluated. The input into the command interpretation loop is taken from console in. Therefore, a clever machine language programmer would be able to have the input to BASIC taken not from the memory of the machine but potentially from a disk file or a cassette file or some other input device, if desired. One of the benefits, if you want to look at it that way, of Color BASIC is that various modules can be added on to BASIC very easily. That is why it is possible to buy extended BASIC, plug it into the computer, turn it on and have it work with no hardware modification. Disk BASIC may be added and provision has even been given for a user add-on. The primary and secondary tokens have a dictionary table, which is the ASCII representation of the various commands and functions spelled out in the memory of the computer. The last byte of the command or function, such as PRINT or LIST, has bit 7 set. This is done so that BASIC can look through the lists of commands and functions and know when a particular command or function spelling is done. In this way an input command in a BASIC program is matched against the command, which is stored in the computer's memory. The computer knows if Extended BASIC, Disk BASIC or a user command table has been implemented. Therefore, it will search through the list of BASIC commands and if it does not find the command that you have typed in the list of BASIC commands, it will then go to Extended BASIC's command table. If it doesn't find it there, it will go to Disk BASIC's command table. If it doesn't find it there it will go to the User Supplied Command Table, if one has been given. When a command is found, there is an associated address in the dispatch table, which tells BASIC where control has to be transferred in order

to perform the various functions, which have been called from the BASIC program. These linkable command tables are explained in the memory map of the computer and are located at \$120 (COMVEC). When the first byte of a command table is equal to zero, it is an indication to BASIC that there are no further valid command tables following.

BASIC determines where it is at any one particular time in RAM through the use of the BASIC input pointer. This is an address maintained in the direct page at address \$A5. When BASIC wants to get another character from a BASIC input line in order to determine what function or command needs to be interpreted or to get data or anything else from a BASIC program, it gets this data from the BASIC input pointer. This is done by executing the statement JSR \$9F. This is a small routine which is moved into the direct page from ROM when BASIC is initialized, and when you call this routine it will increment the BASIC input pointer by one, fetch a character from the address pointed to by the BASIC input pointer, load it into accumulator A and then jump back into the main BASIC ROM. The point at which it jumps back into the main BASIC ROM will determine whether the character, which has been fetched from the BASIC program, is numeric. If the character is numeric, the carry flag will be set. Most people are familiar with the Extended BASIC PCLEAR bug, which was caused when a program was written which would PCLEAR memory during the execution of the program. The PCLEAR statement will cause the BASIC program to be moved up or down in RAM if more or fewer pages of graphic RAM are PCLEARED as a result of the statement. Unfortunately, the BASIC program was moved up or down in RAM, but the BASIC input pointer was never moved, therefore the program would be moved to a new place, but the BASIC input pointer would not be moved a corresponding amount. Therefore, BASIC would begin interpreting new program lines from garbage and you would usually get syntax errors.

The stack is used primarily for normal 6809 functions in the Color Computer It does, however, have several auxiliary functions, which it must provide in order to support Color BASIC. For example, when you use a FOR/NEXT loop, 18 bytes of data are stored on the stack so that you can keep track of the index pointer, step value, the initial value that you started the loop at, and the terminal value at which the FOR/NEXT loop will be stopped. All GOSUB return addresses, which are comprised of 5 bytes, are stored on the stack. The expression evaluation routine uses the stack to store many different bytes and floating point numbers on it during the course of the evaluation of an expression. There are several routines in this BASIC and Extended BASIC such as PAINT, BACKUP, COPY, etc., which also use the stack for temporary storage. The experienced machine language programmer will realize that there can be problems with using the stack for temporary storage and variables. If stack storage is used in recursive loops and there are no controls placed upon the limit of the recursive loops, the stack can grow in an uncontrolled manner and will eventually crash into your program and destroy it. For this reason there is a special routine provided in BASIC, which determines if there is enough free RAM to store the amount of data, which you want to store on the stack. This is required so that if you keep storing data on the stack YOU can merely check to verify that there is enough free RAM left in the machine in order to store the data on the stack. If there is not enough free RAM, an OM error results and you exit from the program in a controlled manner. There are, at this time, at least two bugs known to the author, which will create problems during the execution of a program. Both of these bugs are in the Extended BASIC ROM and neither one has been fixed by version 1.1 of Extended BASIC. The first bug is in the PAINT routine, which uses 6 bytes of stack to remember to paint a particular weird angle, which has been left in an odd shaped paint figure. The paint routine only checks to see if there are 4 bytes of free RAM each time it stores 6 bytes of data on the stack. As such, problems can occur in some very weird shaped paint pictures and the stack could actually creep down and either destroy some variables or parts of the BASIC program

and the user would never know what hit him. The second bug is in the PCOPY statement, which will allow the user to PCOPY from page one to page five if only four pages have been PCLEARED. This is interesting and causes a really good explosion that may completely destroy the BASIC program.

VARIABLES

Variables are used by BASIC to keep track of quantities that may take on different values or change during the course of the execution of the program. COLOR BASIC uses only one type of numeric variable, SINGLE PRECISION. Many other BASICs use different types of variables including integer type and double precision. When COLOR BASIC was first developed, the decision was made not to allow double precision or integer type variables because it would take up too much room in the ROM in order to support these variables types. As a result, we have a smaller and more compact ROM but you lose the efficiency and power that is afforded the use of double precision and integer variable types. No matter which type of variable you have, either single precision or string variable, five bytes are allocated for the storage of this variable in the memory of the computer. All variables are identified by a two-character ASCII string, which is the variable name. Variables may have more than two characters in their name but the characters following the second character will always be ignored. This may lead to some confusion when you're writing a BASIC program, but limiting the number of the characters in the variable name to two makes it much more compact and simple to store the variable 'in the memory of the computer. Any useful program has to deal with alphanumeric data. BASIC has a set of functions to deal with these data. Also, all alphanumeric data may be expressed as a continuous connection of characters, which is viewed by BASIC as the value of a single variable.

Color BASIC has a \$ notation which is used to express variables which are strings of alphanumeric data. All of the rules, which apply to normal variables, apply to the string variable.

The limitation on the number of characters that can be stored in a string is 255. The accumulation of characters from an I/O device and the construction of data is accomplished by the concatenation of strings. The operator that is used is +.

Space is allocated for variables only as they are encountered. It is not possible to allocate an array on the basis of 2 single elements; hence, the reason to execute DIM statement before array references. Seven bytes are allocated for each simple variable whether it is a string, number, or user defined function. Each string variable is defined by a five-byte descriptor. This descriptor has five bytes only so that it may be the same length as the single precision floating point variable. Only three of the bytes in the descriptor are actually used to define a string. The other two bytes are wasted but are necessary in order to maintain the some length of descriptor as the floating-point descriptor. Byte zero represents the length of the string and may be any number from 0 to 255. If the length of the string is equal to 0, it indicates a null string. Bytes two and three of the string descriptor are a pointer to the absolute RAM address of the start of the string. When one uses the instruction in BASIC, the access which is returned is the address of the descriptor and this is how you address the string, The absolute RAM address of the string may be anywhere in RAM so that the string may be located in the BASIC program itself, in the string space, or it may even be located in the random buffer file if you have a disk system. If you've been using BASIC for any length of time you may have become familiar with the time delays that occur whenever the BASIC program does what is called housekeeping or garbage collection. What the computer does is to sort all of string space and throws away all of the null strings. This can very often be a long, time-consuming process, which causes the computer to sit and do what appears to be nothing or to be in a hung up state for an extended period of time.

Variables are stored in the variable table, which is immediately following the BASIC program in the memory of the computer. Each variable requires seven bytes to hold its space in the variable table. The first two bytes are the variable name, the next five bytes are the actual value of the variable if it's a floating point number or it's the five byte descriptor if it's a string variable. The variable names as described above contain two ASCII characters. The method that is used to determine whether the number is a floating point variable or a string variable is the condition of the first variable letter. If this first letter has bit seven set, the variable is a string. If bit seven of the first letter of the variable name is not set, that variable is a floating point variable. This is how BASIC determines the type of variable as it searches for a variable in the variable tables. This searching method should be kept in mind when writing BASIC programs so that you can get maximum efficiency and speed out of your program. Variables that are used most often should be located near the top of the table since BASIC starts at the top and works its way to the bottom when it's searching for a variable name. Looking in the BASIC disassembly of expression evaluation, you will find the method that is used in order to find a variable name (LB357). BASIC gets the variable name, which is found by stripping it off of the BASIC line. It then points itself to the beginning of the table and looks all the way through the variable table until it finds a match. If no match is found, then it inserts a variable in the variable table and a value of 0 or a null string if its a string variable is assigned to it. As you can see, this method can be very slow and cumbersome if you have variables that are very often used and are located at the bottom of a very large variable table. Therefore, if at all possible, define the variables, which are to be used most often at the beginning of your program, and this will cause an increase in speed of your BASIC program.

Floating Point Numbers. Single precision variables are stored in the computer as floating point numbers, which are comprised of an exponent, a four-byte mantissa and the sign of the mantissa. In this way, numbers in the approximate range 1E-39 < X < 1E+39 may be saved. A fifth mantissa byte, the sub byte, (FPSBYT), is used in calculations to achieve 9 significant digits of accuracy.

Floating point numbers are always stored with the mantissa "normalized", that is the mantissa is shifted to the left until a "1" is in the high order bit (7) of the most significant byte. When the mantissa sign in not placed into the high order bit of the most significant mantissa byte, the number is "unpacked".

The exponent is computed such that the mantissa $0 = 1 \times 1$. It is stored as a signed 8 bit binary plus a bias of \$80. Negative exponents are not stored 2's complement. Maximum exponent is 10^{38} and minimum exponent is 10^{-39} , which is stored as \$00. A zero exponent is used to flag the number as zero.

Exponent	Approximate	Value
FF	10 ³⁸	
A2	10 ¹⁰	
7F	10-1	
02	10- ³⁶	
00	10-39	

Since the exponent is really a power of 2, it should best be described as the number of left shifts (EXP > \$80) or right shifts (EXP < = \$80) to be performed on the normalized mantissa to create the actual binary representation of the value.

	Example	of	Floating	Point	Numbers	
Expo	nent	MS	MANT	ISSA	LS	Sign
1E38	FF	96	76	99	52	00
4E10	A4	95	02	F9	00	00
2E10	A3	95	02	F9	00	00
1E10	A2	95	02	F9	00	00
1	81	80	00	00	00	00
.5	80	80	00	00	00	00
.25	7F	80	00	00	00	00
1E-4	73	D1	в7	59	59	00
1E-37	06	88	1C	14	14	00
1E-38	02	D9	C7	EE	EE	00
1E-39	00	A0	00	00	00	00
0	00	00	00	00	00	XX
-1	81	80	00	00	00	FF
-10	84	A0	00	00	00	FF

Actual floating point BASIC variables are stored in 5 bytes, rather than 6 bytes as in the floating accumulator. Upon examination, one will note that the most significant byte of the mantissa is always set. If we always assure the number will be in this format, we can use that bit to indicate the sign of the mantissa — thus, freeing the byte used for sign. This is referred to as "packed" format.

The contents of the floating accumulator may be converted to a double byte integer by calling a subroutine INTCNV which is located at \$B3ED The integer is returned in ACCD. An integer can be converted back to floating by loading the two most significant bytes ACCD then calling GIVABF at \$B4F4.

Array Variables. Array variables need not be declared with a DIM statement if they have only one dimension and contain fewer than 10 elements. Each element in an array requires 5 bytes of storage and the format of the 5-byte block is the same as simple variables. Arrays are stored in the array table and each array is preceded by a header block of 5+2*N bytes where N = number of dimensions in the array. The first two bytes contain the name of the array, the next two bytes contain the total length of array items and header block, the fifth byte contains the number of dimensions and, finally, 2 bytes per each dimension contain the length of the dimension.

If large arrays are defined and initialized first before simple variables are assigned, much execution time can be lost moving the arrays each time a simple variable is defined. The best strategy to follow in this case is to assign a value to all known simple variables before assigning arrays. This will optimize execution speed.

CONSOLE INPUT/OUTPUT

Console input and console output are the data channels that are used when transferring information into and out of the computer. There are various different methods that are used by different computer manufacturers in order to control the transmission or the flow of data into and out of the computer. It has become useful in most of the jargon to refer to the process of transferring data into or out of the computer as console input or console output. By using a method such as this, one can merely call the console input function if one wants to get a character in Accumulator A, for instance, and then call the console output device and that character will be placed in the appropriate output device, be it cassette, disk, printer or even the screen. Obviously, something else has to be defined when using console in or console out, such as where we are going to send the character to, or from where we will get the character. The Color computer uses the concept of a device number (DEVNUM) in order to define from where the character is coming or where it is going.

The Color Computer has five different device types associated with it: device number 0 is the screen; -1 is the cassette; -2 is the Line Printer; -3 is the DLOAD (RS232 Download) option and device numbers 1 through 15 represent Disk files. Device number 16 is not accessible to the user because it is used by the system as a temporary scratch disk input/output file. The typical method that one uses to access the console in or console out function is to initially define the device number and then jump to the console input or console output and either get the character back into Accumulator A if you are using console in or to transmit the character to the appropriate output device in Accumulator A if using console out.

CONSOLE INPUT - Get a character from an input buffer, which has been defined somewhere in the computer by the routine, which is being used. Generally speaking, one has to OPEN an input channel with the open command. This is not necessary if one is using the screen because the input from the screen comes from the keyboard, which is always an open channel, and it is not necessary to either open or close it. However, if one is trying to use a cassette file, when the cassette file is open for input, there is a buffer established in the memory of the computer which will allow a block of data to be read from the cassette tape. When the user wants a byte of data out of that buffer, he simply calls console in and the byte is returned from the buffer. In this method the buffer is systematically emptied until, when the last character is taken from the buffer, the computer automatically attempts to read another block of data from the cassette file. If further blocks of data are available, then the buffer is refilled and console in can get more data. If there is no longer any further data in the cassette file, then the EOF flag is set and the user is told that there is no longer any data to be gotten from that device. The same type of method is used with DLOAD and, of course, can't be used from the Line Printer, because the Line Printer is only an output device. The method used to transfer data into and out of Disk files will be explained in the Disk BASIC Unravelled.

CONSOLE OUTPUT - Used to transmit data from the computer to an output device. All of the output devices as defined above may be used for outputting data. The method is very similar to the method used for inputting data from files. For example, if one is using a cassette file, an output buffer is established in the computer's memory. Characters are continuously placed in this output buffer until the buffer is filled with 255 characters. At that time the buffer is flushed, that is, the contents of the buffer are written to tape and further input to the buffer is prohibited until the data block is written to the selected device. Upon completion of the data block transfer, the character buffer is reset to a 0 value,

meaning it is empty, and further data may be input into the buffer. This is the same method that is used by Disk and that method of outputting data to a disk file will be explained in Disk BASIC Unravelled.

It should also be noted that you can not open a DLOAD file for output. That feature has not been implemented in the Color Computer -- DLOAD, can only be used to input data.

0001		8000	EXBAS	EQU	\$8000	
0002		AØØØ	BASIC	EQU	\$AØØØ	
0003		CØØØ	ROMPAK	EQU	\$CØØØ	
0004						
0005		0008	BS	EQU	8	BACKSPACE
0006		ØØØD	CR	EQU	\$D	ENTER KEY
0007		ØØ1B	ESC	EQU	\$1B	ESCAPE CODE
0008		000A	LF	EQU	\$A	LINE FEED
0009		ØØØC	FORMF	EQU	\$C	FORM FEED
0010		0020	SPACE	EQU	\$20	SPACE (BLANK)
0011						,
0012		ØØ3A	STKBUF	EQU	58	STACK BUFFER ROOM
0013		Ø45E	DEBDEL	EQU	\$45E	DEBOUNCE DELAY
0014		ØØFA	LBUFMX	EQU	250	MAX NUMBER OF CHARS IN A BASIC LINE
0015		ØØFA	MAXLIN	EQU	\$FA	MAXIMUM MS BYTE OF LINE NUMBER
ØØ16		DUIA	HANLIN	LQU	ΨIA	MAXIMON NO DITE OF LINE WONDER
ØØ17		2600	DOSBUF	EQU	\$2600	RAM LOAD LOCATION FOR THE DOS COMMAND
ØØ18		0020	DIRLEN	EQU	32	NUMBER OF BYTES IN DIRECTORY ENTRY
ØØ19		0100	SECLEN	EQU	256	LENGTH OF SECTOR IN BYTES
0010		0012	SECMAX	EQU	18	MAXIMUM NUMBER OF SECTORS PER TRACK
ØØ21		1200			SECMAX*SECLEN	LENGTH OF TRACK IN BYTES
			TRKLEN	EQU		
0022		0023	TRKMAX	EQU	35 61(TDVMAY 1)+2	MAX NUMBER OF TRACKS
0023		ØØ4A	FATLEN	EQU	6+(TRKMAX-1)*2	FILE ALLOCATION TABLE LENGTH
0024		0044	GRANMX	EQU	(TRKMAX-1)*2	MAXIMUM NUMBER OF GRANULES
0025		Ø119	FCBLEN	EQU	SECLEN+25	FILE CONTROL BLOCK LENGTH
ØØ26		0010	INPFIL	EQU	\$10	INPUT FILE TYPE
0027		0020	OUTFIL	EQU	\$20	OUTPUT FILE TYPE
0028		0040	RANFIL	EQU	\$40	RANDOM/DIRECT FILE TYPE
0029			+ DOE!!!!!			
0030		~~~	* PSEUDO PSE			
0031		0021	SKP1	EQU	\$21	OP CODE OF BRN SKIP ONE BYTE
0032		ØØ8C	SKP2	EQU	\$8C	OP CODE OF CMPX # - SKIP TWO BYTES
0033		0086	SKP1LD	EQU	\$86	OP CODE OF LDA # - SKIP THE NEXT BYTE
0034			*			AND LOAD THE VALUE OF THAT BYTE INTO ACCA THIS
0035			*			IS USUALLY USED TO LOAD ACCA WITH A NON ZERO VALUE
0036						
0037			* REGISTER A	DDRESSE	S	
0038		FFØØ	PIAØ	EQU	\$FFØØ	PERIPHERAL INPUT ADAPTER #Ø
0039		FF2Ø	PIA1	EQU	\$FF2Ø	PERIPHERAL INPUT ADAPTER #1
0040		FF2Ø	DA	EQU	PIA1+Ø	DIGITAL/ANALOG CONVERTER
0041		FF4Ø	DSKREG	EQU	\$FF4Ø	DISK CONTROL REGISTER
0042		FF48	FDCREG	EQU	\$FF48	1793 CONTROL REGISTER
0043		FFCØ	SAMREG	EQU	\$FFCØ	SAM CONTROL REGISTER
~~ 4 4						
0044						
0044 0045	0000			ORG	Ø	
	0000	0000		ORG SETDP	Ø Ø	
0045	0000	0000				
ØØ45 ØØ46 ØØ47	0000	0000	ENDFLG			STOP/END FLAG: POSITIVE=STOP, NEG=END
ØØ45 ØØ46 ØØ47		0000	ENDFLG CHARAC	SETDP	Ø	STOP/END FLAG: POSITIVE=STOP, NEG=END TERMINATOR FLAG 1
ØØ45 ØØ46 ØØ47 ØØ48	0000	0000		SETDP RMB	Ø 1	
0045 0046 0047 0048 0049	0000 0001	0000	CHARAC	SETDP RMB RMB	Ø 1 1	TERMINATOR FLAG 1
ØØ45 ØØ46 ØØ47 ØØ48 ØØ49 ØØ5Ø	0000 0001 0002	0000	CHARAC ENDCUR	SETDP RMB RMB RMB	Ø 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2
0045 0046 0047 0048 0049 0050 0051	0000 0001 0002 0003	0000	CHARAC ENDCUR TMPLOC	SETDP RMB RMB RMB RMB	Ø 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE
0045 0046 0047 0048 0049 0050 0051 0052	0000 0001 0002 0003 0004	0000	CHARAC ENDCUR TMPLOC IFCTR	RMB RMB RMB RMB RMB	Ø 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING
0045 0046 0047 0048 0049 0050 0051 0052 0053	0000 0001 0002 0003 0004 0005	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG	RMB RMB RMB RMB RMB RMB	0 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054	0000 0001 0002 0003 0004 0005 0006	8888	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP	RMB RMB RMB RMB RMB RMB RMB	0 1 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055	0000 0001 0002 0003 0004 0005 0006 0007	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL	RMB RMB RMB RMB RMB RMB RMB RMB	0 1 1 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055	0000 0001 0002 0003 0004 0005 0006 0007 0008	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS	RMB RMB RMB RMB RMB RMB RMB RMB RMB	0 1 1 1 1 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$ff=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT ←>Ø
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT ◇Ø *TV RELATIONAL OPERATOR FLAG
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT<Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER
0045 0046 0047 0048 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	<pre>0 1 1 1 1 1 1 1 1 1 1 1 2 2</pre>	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS
0045 0046 0047 0048 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000B	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	<pre>0 1 1 1 1 1 1 1 1 1 1 1 2 2 2</pre>	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT<>Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH)
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 0060 0061	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000B 000B	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT<>Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH)
0045 0046 0047 0048 0049 0050 0051 0053 0054 0055 0056 0057 0058 0059 0060 0061	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000B 000D 000F	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$fF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY)
0045 0046 0047 0048 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 0060 0061 0063	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000D 000F 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR T** FLOATING FPA2	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 CCCUMULATOR #2 (MANT	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA
0045 0046 0047 0048 0049 0050 0051 0062 0053 0054 0056 0057 0068 0057 0060 0061 0062 0063	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000D 000F 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 42	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT<>Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK
0045 0046 0047 0048 0049 0050 0051 0052 0053 0055 0056 0057 0058 0059 0060 0061 0062 0063	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000F 0011 0013 0013 0017 0019	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPPT TMPTR1 *** FLOATING FPA2 BOTSTK TXTTAB	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCUMULATOR #2 (MANT 4 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~>Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM
0045 0046 0047 0048 0049 0050 0051 0052 0055 0056 0057 0058 0059 0060 0061 0062 0063 0064 0065	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0000 0000 0000 0000 0000	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB	RMB RMB RMB RMB RMB RMB RMB RMB RMB RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0056 0057 0058 0059 0060 0061 0062 0063 0064	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0009 0001 0001 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT ○Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS
0045 0046 0047 0048 0049 0050 0051 0052 0053 0055 0056 0057 0058 0059 0060 0061 0062 0063 0064 0066 0067 0068	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0000 0001 0001 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYEND FRETOP	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM)
0045 0046 0047 0048 0049 0050 0051 0055 0056 0057 0058 0059 0060 0061 0062 0063 0064 0065 0066 0067	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0008 0001 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYEND FRETOP STRTAB	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING \$PACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT<Ø *TV RELATIONAL OPERATOR FLAG PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM)
0045 0046 0047 0048 0049 0050 0051 0052 0053 0055 0056 0057 0058 0059 0060 0061 0062 0063 0064 0066 0067 0068	0000 0001 0002 0003 0005 0006 0007 0008 0008 0009 0008 0001 0001 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYEND FRETOP	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM)
0045 0046 0047 0048 0049 0050 0051 0052 0055 0055 0056 0057 0068 0061 0062 0063 0064 0065 0066 0067 0066 0067 0069 0069	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0008 0009 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB ARYTAB FRESPC MEMSIZ	RMB	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 CCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE *PV TOP OF STRING SPACE
0045 0046 0047 0048 0049 0050 0051 0052 0055 0056 0057 0059 0060 0061 0062 0063 0064 0067 0068 0067 0068 0067 0068	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0009 0001 0011 0013 0017 0019 0018 0019 0018 0019 0018 0019 0019	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB ARYTAB FRETOP STRTAB FRESPC MEMSIZ OLDTXT	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS *PV END OF ARRAYS *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING VARIABLES UTILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP"
0045 0046 0047 0048 0049 0050 0051 0055 0056 0057 0058 0060 0061 0063 0064 0065 0066 0067 0068 0067 0068 0067 0068	0000 0001 0002 0003 0006 0007 0006 0007 0008 0009 0000 0001 0011 0013 0011 0019 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYEND FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL	RMB	0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT<Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER
0045 0046 0047 0048 0049 0050 0051 0052 0053 0055 0056 0057 0068 0061 0062 0063 0066 0067 0068 0067 0069 0070 0071	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0000 0001 0011 0013 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR	RMB	0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT ◇Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE UTILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED INPUT PTR DURING A "STOP"
0045 0046 0047 0048 0049 0050 0051 0052 0053 0055 0056 0057 0068 0060 0061 0066 0067 0068 0069 0071 0073 0074 0075 0075	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0001 0011 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB ARYTAB FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	Ø 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT PUDINTER STOPP" TEMPORARY INPUT POINTER STOPP"
0045 0046 0047 0048 0049 0050 0051 0052 0055 0055 0056 0057 0058 0064 0067 0068 0067 0068 0067 0072 0073 0074 0075	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0008 0001 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTXT	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER *TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS *PV END OF ARRAYS *PV END OF ARRAYS *PV END OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED INPUT PTR DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT LINE NUMBER POINTER
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0056 0057 0068 0061 0062 0063 0064 0066 0067 0068 0069 0070 0072 0073 0076 0077	00000 00011 0002 0003 00005 00006 00007 00008 00009 00008 00001 00011 00113 00117 00119 00118 00118 00119 00	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTAB VARTAB ARYTAB ARYTAB ARYEND FETENPE FESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTYT DATPTR	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT >Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE UTILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT LINE NUMBER POINTER
0045 0046 0047 0048 0049 0050 0051 0052 0053 0055 0056 0057 0068 0067 0066 0067 0068 0067 0069 0071 0072 0073 0074 0077 0078	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0001 0011 0013 0019 0019 0019 0019	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB ARYTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTXT DATTTR DATTMP	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV' DATA' STATEMENT LINE NUMBER POINTER *PV' DATA' STATEMENT LINE NUMBER POINTER *PV' OATA' STATEMENT LINE NUMBER POINTER *PATA POINTER FOR 'INPUT' & 'READ'
0045 0046 0047 0048 0049 0050 0051 0052 0053 0056 0057 0058 0066 0067 0068 0067 0068 0067 0071 0072 0073 0074 0075 0077 0078 0079 0081	0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 0008 0001 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB ARYTAB FRESPC MEMSIZ OLDTXT BINVAL OLDTXT BINVAL OLDTR DATTXT DATTR DATTTR DATTTR DATTTR DATTRD	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT ADDRESS POINTER *PV 'DATA' STATEMENT ADDRESS POINTER *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT LINE NUMBER POINTER *TV TEMP STORAGE FOR A VARIABLE NAME
0045 0046 0047 0048 0049 0050 0051 0055 0056 0057 0058 0056 0067 0068 0067 0068 0067 0068 0067 0070 0070	00000 00011 0002 0003 0006 0006 0007 0008 0000 00011 0011 0011 0011 001	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYEND FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTXT DATTMP VARNAM VARPTR	RMB	0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT >Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER *TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT ADDRESS POINTER *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT ADDRESS POINTER DATA POINTER FOR 'INPUT' & 'READ' *TV TEMP STORAGE FOR A VARIABLE NAME *TV POINTER TO A VARIABLE DESCRIPTOR
0045 0046 0047 0048 0049 0050 0051 0052 0053 0054 0055 0066 0057 0068 0061 0062 0063 0064 0067 0068 0067 0068 0067 0068 0067 0070 0070	00000 00001 00002 00003 00005 00006 00007 00008 00008 00001 00001 00010 0011 00119 0	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTAB VARTAB ARYEND FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTYT DATTYR DATTYR VARNAM VARPTR VARNAM VARPTR VARDES	RMB	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT-ØØ *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING VARIABLES UTLILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED INPUT PTR DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT ADDRESS POINTER DATA POINTER FOR 'INPUT' & 'READ' *TV TEMP STORAGE FOR A VARIABLE BAME *TV POINTER TO A VARIABLE DESCRIPTOR TEMP POINTER TO A VARIABLE DESCRIPTOR
0045 0046 0047 0048 0049 0050 0051 0052 0053 0055 0056 0057 0068 0061 0062 0063 0064 0065 0067 0068 0067 0068 0070 0071 0072 0073 0074 0075 0076 0077 0078 0077	0000 0001 0002 0003 0006 0006 0006 0007 0008 0007 0011 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYTAB ARYTAB FRETOP STRTAB FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTXT DATTTR DATTXT DATTMP VARNAM VARPTR VARDES RELPTR	SETDP RMB RMB RMB RMB RMB RMB RMB RMB RMB RM	Ø 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT~Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING VARIABLES UTILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT ADDRESS POINTER ADTA POINTER FOR 'INPUT' & "READ' *TV TEMP STORAGE FOR A VARIABLE NAME *TV POINTER TO A VARIABLE DESCRIPTOR POINTER TO RELATIONAL OPERATOR PROCESSING ROUTINE
0045 0046 0047 0048 0049 0050 0051 0052 0053 0056 0057 0058 0060 0061 0062 0063 0064 0067 0068 0067 0068 0067 0071 0072 0073 0074 0075 0077 0078 0079 00881 0082 0083	0000 0001 0002 0003 0006 0006 0006 0007 0008 0007 0011 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR TMPTR1 ** FLOATING FPA2 BOTSTK TXTAB VARTAB ARYEND FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTYT DATTYR DATTYR VARNAM VARPTR VARNAM VARPTR VARDES	RMB	Ø 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT-ØØ *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER TEMPORARY POINTER TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING VARIABLES UTLILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED INPUT PTR DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT ADDRESS POINTER DATA POINTER FOR 'INPUT' & 'READ' *TV TEMP STORAGE FOR A VARIABLE BAME *TV POINTER TO A VARIABLE DESCRIPTOR TEMP POINTER TO A VARIABLE DESCRIPTOR
0045 0046 0047 0048 0049 0050 0051 0052 0055 0056 0057 0058 0059 0060 0061 0062 0063 0064 0067 0068 0067 0068 0077 0078 0077 0078 0077 0077	0000 0001 0002 0003 0006 0006 0006 0007 0008 0007 0011 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYEND FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTXT DATTMP VARNAM VARPTR VARDES RELPTR TRELFL	RMB	0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT > Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER *TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE UTILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT LINE NUMBER NAME *TV POINTER TO A VARIABLE DESCRIPTOR TEMP POINTER TO A VARIABLE DESCRIPTOR TEMP POINTER TO A VARIABLE DESCRIPTOR TEMPORARY RELATIONAL OPERATOR FRAGE BYTE
0045 0046 0047 0048 0049 0050 0051 0052 0053 0056 0057 0058 0060 0061 0062 0063 0064 0067 0068 0067 0068 0067 0071 0072 0073 0074 0075 0077 0078 0079 00881 0082 0083	0000 0001 0002 0003 0006 0006 0006 0007 0008 0007 0011 0011 0011 0011 0011	0000	CHARAC ENDCUR TMPLOC IFCTR DIMFLG VALTYP GARBFL ARYDIS INPFLG RELFLG TEMPPT LASTPT TEMPTR ** FLOATING FPA2 BOTSTK TXTTAB VARTAB ARYTAB ARYTAB ARYTAB ARYEND FRETOP STRTAB FRESPC MEMSIZ OLDTXT BINVAL OLDPTR TINPTR DATTXT DATTMP VARNAM VARPTR VARDES RELPTR TRELFL	RMB	Ø 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 CCCUMULATOR #2 (MANT 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	TERMINATOR FLAG 1 TERMINATOR FLAG 2 SCRATCH VARIABLE IF COUNTER - HOW MANY IF STATEMENTS IN A LINE *DV* ARRAY FLAG Ø=EVALUATE, 1=DIMENSIONING *DV* *PV TYPE FLAG: Ø=NUMERIC, \$FF=STRING *TV STRING SPACE HOUSEKEEPING FLAG DISABLE ARRAY SEARCH: ØØ=ALLOW SEARCH *TV INPUT FLAG: READ=Ø, INPUT > Ø *TV RELATIONAL OPERATOR FLAG *PV TEMPORARY STRING STACK POINTER *PV ADDR OF LAST USED STRING STACK ADDRESS TEMPORARY POINTER *TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH) ISSA ONLY) FLOATING POINT ACCUMULATOR #2 MANTISSA BOTTOM OF STACK AT LAST CHECK *PV BEGINNING OF BASIC PROGRAM *PV START OF VARIABLES *PV START OF ARRAYS *PV END OF ARRAYS (+1) *PV START OF STRING STORAGE (TOP OF FREE RAM) *PV START OF STRING STORAGE UTILITY STRING POINTER *PV TOP OF STRING SPACE SAVED LINE NUMBER DURING A "STOP" BINARY VALUE OF A CONVERTED LINE NUMBER SAVED LINE NUMBER DURING A "STOP" TEMPORARY INPUT POINTER STORAGE *PV 'DATA' STATEMENT LINE NUMBER POINTER *PV 'DATA' STATEMENT LINE NUMBER NAME *TV POINTER TO A VARIABLE DESCRIPTOR TEMP POINTER TO A VARIABLE DESCRIPTOR TEMP POINTER TO A VARIABLE DESCRIPTOR TEMPORARY RELATIONAL OPERATOR FRAGE BYTE

~~~~							
0088		* USED AS SC				ED. (#40	***
ØØ89	aa4a	** FLOATING			#3 :PACKI	ED: (\$40-	- \$44)
0090		V4Ø	RMB	1			
0091		V41	RMB	1			
0092		V42	RMB	1			
0093		V43	RMB	1			
0094	0044	V44	RMB	1		/	***
0095		** FLOATING			#4 : PACKI	ED: (\$45-	-\$49)
0096		V45	RMB	1			
ØØ97		V46	RMB	1			
ØØ98		V47	RMB	1			
0099	0048	V48	RMB	2			
0100		** FLOATING			#5 : PACK	ED: (\$4A	\$4E)
0101	ØØ4A	V4A	RMB	1			
0102	ØØ4B	V4B	RMB	2			
0103	ØØ4D	V4D	RMB	2			
0104		** FLOATING	POINT	ACCUMULATOR	#Ø		
0105	004F	FPØEXP	RMB	1		*PV FLOA	ATING POINT ACCUMULATOR #Ø EXPONENT
0106	0050	FPAØ	RMB	4		*PV FLOA	ATING POINT ACCUMULATOR #Ø MANTISSA
0107	0054	FPØSGN	RMB	1		*PV FLOA	ATING POINT ACCUMULATOR #Ø SIGN
0108	0055	COEFCT	RMB	1		POLYNOM	IAL COEFFICIENT COUNTER
0109	0056	STRDES	RMB	5		TEMPORAR	RY STRING DESCRIPTOR
0110	ØØ5B	FPCARY	RMB	1		FLOATING	POINT CARRY BYTE
Ø111		** FLOATING			#1		
Ø112	ØØ5C	FP1EXP	RMB	1		*PV FLOA	ATING POINT ACCUMULATOR #1 EXPONENT
Ø113		FPA1	RMB	4			ATING POINT ACCUMULATOR #1 MANTISSA
Ø114		FP1SGN	RMB	1			ATING POINT ACCUMULATOR #1 SIGN
Ø115				_			
Ø116	0062	RESSGN	RMB	1		SIGN OF	RESULT OF FLOATING POINT OPERATION
Ø117		FPSBYT	RMB	1			POINT SUB BYTE (FIFTH BYTE)
Ø117		COEFPT	RMB	2			IAL COEFFICIENT POINTER
Ø119		LSTTXT	RMB	2			LINE POINTER DURING LIST
0120		CURLIN	RMB	2			RENT LINE # OF BASIC PROGRAM, \$FFFF = DIRECT
Ø121		DEVCFW	RMB	1			FIELD WIDTH
Ø122		DEVLCF	RMB	1		*TV TAB	
Ø123		DEVPOS	RMB	1			NT POSITION
0124		DEVWID	RMB	1		*TV PRIM	
Ø125		PRTDEV	RMB	1			NT DEVICE: Ø=NOT CASSETTE, -1=CASSETTE
Ø126	ØØ6F	DEVNUM	RMB	1		*PV DEVI	ICE NUMBER: -3=DLOAD, -2=PRINTER,
Ø127		*					-1=CASSETTE, Ø=SCREEN, 1-15=DISK
Ø128	0070	CINBFL	RMB	1		*PV CONS	SOLE IN BUFFER FLAG: ØØ=NOT EMPTY, \$FF=EMPTY
Ø129	0071	RSTFLG	RMB	1		*PV WARN	1 START FLAG: \$55=WARM, OTHER=COLD
Ø13Ø	0072	RSTVEC	RMB	2		*PV WARM	1 START VECTOR - JUMP ADDRESS FOR WARM START
Ø131	0074	TOPRAM	RMB	2		*PV TOP	OF RAM
Ø132	0076		RMB	2		SPARE: U	JNUSED VARIABLES
Ø133	0078	FILSTA	RMB	1		*PV FILE	STATUS FLAG: Ø=CLOSED, 1=INPUT, 2=OUTPUT
Ø134	ØØ79	CINCTR	RMB	1			SOLE IN BUFFER CHAR COUNTER
Ø135		CINPTR	RMB	2			SOLE IN BUFFER POINTER
Ø136		BLKTYP	RMB	1			S BLOCK TYPE: Ø=HEADER, 1=DATA, \$FF=EOF
Ø137		BLKLEN	RMB	1			SETTE BYTE COUNT
Ø138		CBUFAD	RMB	2			SETTE LOAD BUFFER POINTER
Ø139		CCKSUM	RMB	1			SETTE CHECKSUM BYTE
Ø14Ø		CSRERR	RMB	1			OR FLAG/CHARACTER COUNT
Ø141			RMB				SE WIDTH COUNT
Ø141		CPULWD CPERTM	RMB	1 1		*TV BIT	
Ø142			RMB				PHASE FLAG
		CBTPHA	RMB	1			
Ø144		CLSTSN		1			SINE TABLE ENTRY
Ø145		GRBLOK	RMB	1			PHIC BLOCK VALUE FOR SET, RESET AND POINT
Ø146		IKEYIM	RMB	1			EY\$ RAM IMAGE
Ø147		CURPOS	RMB	2			SOR LOCATION
Ø148		ZERO	RMB	2			MY - THESE TWO BYTES ARE ALWAYS ZERO
Ø149		SNDTON	RMB	1			E VALUE FOR SOUND COMMAND
0150	ดดลก	SNDDUR	RMB	2		*IV DURA	ATION VALUE FOR SOUND COMMAND
0151							
Ø152		** THESE BYT	ES ARE	MOVED DOWN	FROM ROM		
Ø153		***				INIT	DESCRIPTION
Ø154		*				VALUE	
Ø155		CMPMID	RMB	1		18	*PV 1200/2400 HERTZ PARTITION
Ø156		CMPØ	RMB	1		24	*PV UPPER LIMIT OF 1200 HERTZ PERIOD
Ø157		CMP1	RMB	1		10	*PV UPPER LIMIT OF 2400 HERTZ PERIOD
Ø158	0092	SYNCLN	RMB	2		128	*PV NUMBER OF \$55'S TO CASSETTE LEADER
Ø159	0094	BLKCNT	RMB	1		11	*PV CURSOR BLINK DELAY
0160	0095	LPTBTD	RMB	2		88	*PV BAUD RATE CONSTANT (600)
Ø161		LPTLND	RMB	2		1	*PV PRINTER CARRIAGE RETURN DELAY
Ø162		LPTCFW	RMB	1		16	*PV TAB FIELD WIDTH
Ø163		LPTLCF	RMB	1		112	*PV LAST TAB ZONE
Ø164		LPTWID	RMB	1		132	*PV PRINTER WIDTH
Ø165		LPTPOS	RMB	1		Ø	*PV LINE PRINTER POSITION
Ø166		EXECJP	RMB	2			*PV JUMP ADDRESS FOR EXEC COMMAND
Ø167	2230	LALOU!	KIID	_		ニレ・ハハ	John Applicas For Exec Command
Ø168		** THIS ROUT	TNF DT	CKS IID THE	NEXT INDII	T CHADACT	FFR FROM
Ø169		** BASIC. TH					
						וום ל	, DE
Ø17Ø		** INTERPRET	ED 12	SIUKED AT C	пикаи.		
Ø171	000F 00 A7	CETNOU	THO	-CHADAD : -		+DV 7***	DEMENT IS DATE OF THRUT DOTHES
	009F 0C A7	GETNCH	INC	<charad+1< td=""><td></td><td></td><td>REMENT LS BYTE OF INPUT POINTER</td></charad+1<>			REMENT LS BYTE OF INPUT POINTER
	00A1 26 02		BNE	GETCCH			NCH IF NOT ZERO (NO CARRY)
Ø1/4	ØØA3 ØC A6		INC	<charad< td=""><td></td><td>arv INCh</td><td>REMENT MS BYTE OF INPUT POINTER</td></charad<>		arv INCh	REMENT MS BYTE OF INPUT POINTER

Ø175	ØØA5	B6	GETCCH	FCB	\$B6	*PV OP CODE OF LDA EXTENDED
Ø176	ØØA6		CHARAD		2	*PV THESE 2 BYTES CONTAIN ADDRESS OF THE CURRENT
Ø177			*			CHARACTER WHICH THE BASIC INTERPRETER IS
Ø178			*			PROCESSING
Ø179	ØØA8	7E AA 1A		JMP	BROMHK	JUMP BACK INTO THE BASIC RUM
0180						
Ø181	ØØAB		VAB	RMB	1	= LOW ORDER FOUR BYTES OF THE PRODUCT
Ø182	ØØAC		VAC	RMB	1	= OF A FLOATING POINT MULTIPLICATION
Ø183	ØØAD		VAD	RMB	1	= THESE BYTES ARE USE AS RANDOM DATA
Ø184	ØØAE		VAE	RMB	1	= BY THE RND STATEMENT
Ø185						
Ø186			* EXTENDED B	ASIC VA	RIABLES	
Ø187	ØØAF		TRCFLG	RMB	1	*PV TRACE FLAG Ø=OFF ELSE=ON
Ø188	ØØBØ		USRADR	RMB	2	*PV ADDRESS OF THE START OF USR VECTORS
Ø189			FORCOL	RMB	1	*PV FOREGROUND COLOR
0190			BAKCOL	RMB	1	*PV BACKGROUND COLOR
Ø191			WCOLOR	RMB	1	*TV WORKING COLOR BEING USED BY EX BASIC
Ø192			ALLCOL	RMB	1	*TV ALL PIXELS IN THIS BYTE SET TO COLOR OF VB3
Ø193			PMODE	RMB	1	*PV PMODE'S MODE ARGUMENT
Ø194			ENDGRP	RMB	2	*PV END OF CURRENT GRAPHIC PAGE
Ø195			HORBYT	RMB	1	*PV NUMBER OF BYTES/HORIZONTAL GRAPHIC LINE
Ø196			BEGGRP	RMB	2	*PV START OF CURRENT GRAPHIC PAGE
Ø197			GRPRAM	RMB	1	*PV START OF GRAPHIC RAM (MS BYTE)
Ø198			HORBEG	RMB	2	*DV* *PV HORIZ COORD - START POINT
Ø199			VERBEG	RMB	2	*DV* *PV VERT COORD - START POINT
0200			CSSYAL	RMB	1	*PV SCREEN'S COLOR SET ARGUMENT
0201			SETFLG	RMB	1	*PV PRESET/PSET FLAG: Ø=PRESET, 1=PSET
0202			HOREND	RMB	2	*DV* *PV HORIZ COORD - ENDING POINT
Ø2Ø3 Ø2Ø4			VEREND HORDEF	RMB	2	*DV* *PV VERT COORD - ENDING POINT
				RMB	2	*PV HORIZ COORD - DEFAULT COORD
Ø2Ø5	иис9		VERDEF	RMB	2	*PV VERT COORD - DEFAULT COORD
Ø2Ø6 Ø2Ø7			+ FVTENDED D	ACTC CC	DATCH DAD VADIADIES	
Ø2Ø8	aacp		VCB	RMB	CRATCH PAD VARIABLES 2	
Ø2Ø9			VCD	RMB	2	
Ø21Ø			VCF	RMB	2	
Ø210			VD1	RMB	2	
Ø211			VD3	RMB	1	
Ø212			VD3 VD4	RMB	1	
Ø213			VD5	RMB	1	
Ø215			VD6	RMB	1	
Ø216			VD7	RMB	1	
Ø217			VD8	RMB	1	
Ø218			VD9	RMB	1	
Ø219			VDA	RMB	1	
0220	22211			5	-	
Ø221	ØØDB		CHGFLG	RMB	1	*TV FLAG TO INDICATE IF GRAPHIC DATA HAS BEEN CHANGED
Ø222			TMPSTK	RMB	2	*TV STACK POINTER STORAGE DURING PAINT
Ø223			OCTAVE	RMB	1	*PV OCTAVE VALUE (PLAY)
Ø224	ØØDF		VOLHI	RMB	1	*DV* *PV VOLUME HIGH VALUE (PLAY)
Ø225	ØØEØ		VOLLOW	RMB	1	*DV* *PV VOLUME LOW VALUE (PLAY)
Ø226	ØØE1		NOTELN	RMB	1	*PV NOTE LENGTH (PLAY)
Ø227	ØØE2		TEMP0	RMB	1	*PV TEMPO VALUE (PLAY)
Ø228	ØØE3		PLYTMR	RMB	2	*TV TIMER FOR THE PLAY COMMAND
Ø229	ØØE5		DOTYAL	RMB	1	*TV DOTTED NOTE TIMER SCALE FACTOR
0230	ØØE6		DLBAUD	RMB	1	*DV* *PV DLOAD BAUD RATE CONSTANT \$BØ=300, \$2C=1200
Ø231	ØØE7		TIMOUT	RMB	1	*DV* *PV DLOAD TIMEOUT CONSTANT
Ø232	ØØE8		ANGLE	RMB	1	*DV* *PV ANGLE VALUE (DRAW)
Ø233	ØØE9		SCALE	RMB	1	*DV* *PV SCALE VALUE (DRAW)
Ø234						
Ø235			* DSKCON VAR			
Ø236			DCOPC	RMB	1	*PV DSKCON OPERATION CODE Ø-3
Ø237			DCDRV	RMB	1	*PV DSKCON DRIVE NUMBER Ø 3
Ø238			DCTRK	RMB	1	*PV DSKCON TRACK NUMBER Ø 34
Ø239			DSEC	RMB	1	*PV DSKCON SECTOR NUMBER 1-18
0240			DCBPT	RMB	2	*PV DSKCON DATA POINTER
Ø241	מושמ		DCSTA	RMB	1	*PV DSKCON STATUS BYTE
Ø242 Ø243	00F1		FCBTMP	RMB	2	TEMPORARY FCB POINTER
Ø243 Ø244	דושש		COIMP	VIID	_	TERRORAKI FUD FUINIEK
Ø244 Ø245	aars			RMB	13	SPARE: UNUSED VARIABLES
Ø245 Ø246	בחשש			KIID	10	STARE, UNUSED VARIABLES
Ø240						
Ø247			*			BASIC EXBASIC DOSBASIC
Ø249						
Ø25Ø	ดาดด		SW3VEC	RMB	3	\$XXXX \$XXXX \$3B3B SWI3 VECTOR
Ø250			SW2VEC	RMB	3	\$XXXX \$XXXX \$3B3B SWI2 VECTOR
Ø251			SWIVEC	RMB	3	\$XXXX \$XXXX \$XXXX SWI VECTOR
Ø253			NMIVEC	RMB	3	\$XXXX \$XXXX \$D7AE NMI VECTOR
Ø254			IRQVEC	RMB	3	\$A9B3 \$894C \$D7BC IRQ VECTOR
Ø255			FRQVEC	RMB	3	\$AØF6 \$AØF6 \$AØF6 FIRQ VECTOR
Ø256						
Ø257	Ø112		TIMVAL			
Ø258			USRJMP	RMB	3	JUMP ADDRESS FOR BASIC'S USR FUNCTION
Ø259			*	RMB	2	TIMER VALUE FOR EXBAS
0260			*	RMB	1	UNUSED BY EXBAS OR DISK BASIC
0261	0115		RVSEED	RMB	1	* FLOATING POINT RANDOM NUMBER SEED EXPONENT

Ø262			RMB	4	* MANTISSA: INITIALLY SET TO \$804FC75259
Ø263		CASFLG	RMB	1	UPPER CASE/LOWER CASE FLAG: \$FF=UPPER, Ø=LOWER
Ø264		DEBVAL	RMB	2	KEYBOARD DEBOUNCE DELAY (SET TO \$45E)
Ø265	Ø11D	EXPJMP **	RMB	3	JUMP ADDRESS FOR EXPONENTIATION
Ø266 Ø267		^^			INITIALLY SET TO ERROR FOR BASIC, \$8489 FOR EX BASIC
Ø268		***	COMMAN	D INTERPRETATION VEC	TOP TARIF
Ø269			COMMAN	D INTERCRETATION VEC	TOK TABLE
Ø27Ø		** FOUR SETS	OF 10	RYTE TARIES:	
Ø271		10011 0210	0. 10	5112 INSEE 01	
Ø272					
Ø273		** THE LAST	USED TA	BLE MUST BE FOLLOWE	D BY A ZERO BYTE
Ø274		* THE JUMP T	ABLE VE	CTORS (3,4 AND 8,9)	POINT TO THE JUMP TABLE FOR
Ø275		* THE FIRST	TABLE.	FOR ALL OTHER TABLE	S, THESE VECTORS POINT TO A
Ø276		* ROUTINE WH	ICH WIL	L VECTOR YOU TO THE	CORRECT JUMP TABLE.
Ø277					IS SCHEME SO THAT THE USER
Ø278					TIONAL TABLES WILL HAVE TO BE
Ø279		* ACCESSED F	ROM A N	EW COMMAND HANDLER.	
0280			DVTE	DECODIDATION	
Ø281		*	BYTE	DESCRIPTION	HODDC
Ø282 Ø283		*	Ø 1,2	NUMBER OF RESERVED LOOKUP TABLE OF RES	
Ø284		*	3,4	JUMP TABLE FOR COMM	
Ø285		*	5,4		COMMAND HANDLERS (ALL BUT FIRST TABLE)
Ø286		*	5	NUMBER OF SECONDARY	
Ø287		*	6.7		ONDARY FUNCTIONS (FIRST TABLE)
Ø288		*	,	VECTOR TO EXPANSION	SECONDARY COMMAND HANDLERS (ALL BUT
Ø289		*		FIRST TABLE)	
0290		*	8,9	JUMP TABLE FOR SECO	NDARY FUNCTIONS
0291		*	10	Ø BYTE - END OF TAE	LE FLAG (LAST TABLE ONLY)
Ø292					
Ø293	0120	COMVEC	RMB	10	BASIC'S TABLE
Ø294			RMB	10	EX BASIC'S TABLE
Ø295	0134		RMB	10	DISC BASIC'S TABLE (UNUSED BY EX BASIC)
Ø296					
Ø297	710F	**** USR FUN		ECTOR ADDRESSES (EX	
	Ø13E		RMB	2	USR Ø VECTOR
Ø299	0140 0142		RMB RMB	2	USR 1 USR 2
Ø3Ø1			RMB	2	USR 3
Ø3Ø2			RMB	2	USR 4
Ø3Ø3			RMB	2	USR 5
0304			RMB	2	USR 6
Ø3Ø5			RMB	2	USR 7
Ø3Ø6			RMB	2	USR 8
Ø3Ø7			RMB	2	USR 9
0308					
0309		*** THE ABOV	E 20 BY	TE USR ADDR VECTOR	TABLE IS MOVED TO
Ø31Ø		*** \$95F-\$97	2 BY DI	SC BASIC. THE 20 BY	TES FROM \$13E-\$151
Ø311		*** ARE REDE	FINED A	S FOLLOWS:	
Ø312					
Ø313		*	RMB	10	USER (SPARE) COMMAND INTERPRETATION TABLE SPACE
Ø314		*	FCB	Ø	END OF COMM INTERP TABLE FLAG
Ø315		*	RMB	9	UNUSED BY DISK BASIC
Ø316		*	COMMAN	D INTERDRETATION TAG	N.E. VALUEC
Ø317		*	CUMMAN	D INTERPRETATION TAE BYTE	BASIC EX BASIC DISK BASIC
Ø318 Ø319		*		Ø	53 BASIC TABLE
Ø32Ø		*		1,2	\$AA66
Ø320		*		3,4	\$AB67
Ø322		*		5	20
Ø323		*		6,7	\$AB1A
Ø324		*		8,9	\$AA29
Ø325					
Ø326		*		Ø	25 EX BASIC TABLE
Ø327		*		1,2	\$8183
Ø328		*		3,4	\$813C \$CE2E (\$CFØA 2.1)
Ø329		*		5	14
Ø33Ø		*		6,7	\$821E
Ø331		*		8,9	\$8168 \$CE56 (\$CF32 2.1)
Ø332		*		α	10 (20 2 1) DICK DACIC TABLE
Ø333 Ø334		*		Ø 1,2	19 (20 2.1) DISK BASIC TABLE \$C17F
Ø335		*		3,4	\$C2CØ
Ø336		*		5	6
Ø337		*		6,7	\$C2Ø1
Ø338		*		8,9	\$C236
Ø339				•	
0340					
	Ø152	KEYBUF	RMB	8	KEYBOARD MEMORY BUFFER
	Ø15A	POTVAL	RMB	1	LEFT VERTICAL JOYSTICK DATA
Ø343	Ø15B		RMB	1	LEFT HORIZONTAL JOYSTICK DATA
Ø344	Ø15C		RMB	1	RIGHT VERTICAL JOYSTICK DATA
Ø345	Ø15D		RMB	1	RIGHT HORIZONTAL JOYSTICK DATA
Ø346					
Ø347				RS - INITIALIZED TO	
Ø348		* 25 SETS OF	3 BYTE	INSTRUCTIONS WHICH	ARE CALLED BY COLOR BASIC

Ø349				BASIC. THEIR PURPO				TS (SUCH			
Ø35Ø		* AS EX BASIC AND DOS BASIC) AS MORE ROMS ARE ADDED TO THE  * SYSTEM BY FEFETIVELY ALLOWING MODE CODE TO BE ADDED TO THE									
Ø351		* SYSTEM BY EFFECTIVELY ALLOWING MORE CODE TO BE ADDED TO THE  * ROUTINES IN EARLIER ROMS. THIS NEW CODE IS LOCATED IN THE NEW ROMS									
Ø352											
Ø353		* AND THE ADDRESS TO GET TO THE NEW CODE IS IN BYTES 1 & 2 OF THE * RAM VECTOR. BYTE Ø WILL CONTAIN A \$7E WHICH IS THE FIRST BYTE OF									
Ø354					F MHICH	IS THE F.	IKZI BALE	. OF			
Ø355		* THE JMP INSTRUCTION.  * THE FIRST ADDRESS IN THIS TABLE IS THE ADDRESS IN BASIC WHICH									
Ø356											
Ø357				TOR, THE SECOND ADD							
Ø358				THE RAM VECTOR (IF			RD ADDRES	S			
Ø359		* IS THE VAL	UE WHIC	CH DISK BASIC PUTS T	HERE (IF	ANY)					
Ø36Ø											
Ø361											
Ø362		*			2.0	2.1	1.0	1.1			
Ø363	Ø15E	RVECØ	RMB	3	\$A5F6		\$C426	\$C44B	OPEN COMMAND		
Ø364	Ø161	RVEC1	RMB	3	\$A5B9		\$C838	\$C888	DEVICE NUMBER VALIDITY CHECK		
Ø365	Ø164	RVEC2	RMB	3	\$A35F		\$C843	\$C893	SET PRINT PARAMETERS		
Ø366	Ø167	RVEC3	RMB	3	\$A282	\$8273	\$CB4A	\$CC1C	CONSOLE OUT		
Ø367		RVEC4	RMB	3	\$A176	\$8CF1	\$C58F	\$C5BC	CONSOLE IN		
Ø368		RVEC5	RMB	3	\$A3ED		\$C818	\$C848	INPUT DEVICE NUMBER CHECK		
Ø369		RVEC6	RMB	3	\$A4Ø6		\$C81B	\$C84B	PRINT DEVICE NUMBER CHECK		
Ø37Ø		RVEC7	RMB	3	\$A426		\$CA3B	\$CAE9	CLOSE ALL FILES		
Ø378		RVEC8	RMB	3	\$A42D	\$8286	\$CA3B	\$CAF9	CLOSE ONE FILE		
Ø371		RVEC9	RMB	3	\$B918	\$8E9Ø	\$8E9Ø	\$8E9Ø	PRINT		
Ø372						40E30					
		RVEC1Ø	RMB	3	\$BØ61		\$CC5B	\$CD35	INPUT		
Ø374		RVEC11	RMB	3	\$A549		\$C859	\$C8A9	BREAK CHECK		
Ø375		RVEC12	RMB	3	\$A39Ø		\$C6B7	\$C6E4	INPUTTING A BASIC LINE		
Ø376		RVEC13	RMB	3	\$A4BF		\$CA36	\$CAE4	TERMINATING BASIC LINE INPUT		
Ø377		RVEC14	RMB	3	\$A5CE		\$CA6Ø	\$C9ØC	EOF COMMAND		
Ø378		RVEC15	RMB	3	\$B223	\$8846	\$CDF6	\$CED2	EVALUATE AN EXPRESSION		
Ø379	Ø18E	RVEC16	RMB	3	\$AC46		\$C6B7	\$C6E4	RESERVED FOR ON ERROR GOTO COMMAND		
Ø38Ø	0191	RVEC17	RMB	3	\$AC49	\$88FØ	\$C24D	\$C265	ERROR DRIVER		
Ø381	Ø194	RVEC18	RMB	3	\$AE75	\$829C	\$C99Ø	\$CA3E	RUN		
Ø382	Ø197	RVEC19	RMB	3	\$BD22	\$87EF			ASCII TO FLOATING POINT CONVERSION		
Ø383	Ø19A	RVEC2Ø	RMB	3	\$AD9E	\$82B9		\$C8BØ	BASIC'S COMMAND INTERPRETATION LOOP		
Ø384	Ø19D	RVEC21	RMB	3	\$A8C4				RESET/SET/POINT COMMANDS		
Ø385	Ø1AØ	RVEC22	RMB	3	\$A910				CLS		
Ø386		*			\$8162				EXBAS' SECONDARY TOKEN HANDLER		
Ø387		*			\$8AFA				EXBAS' RENUM TOKEN CHECK		
Ø388		*			\$975C		\$C29A	\$C2B2	EXBAS' GET/PUT		
Ø389	Ø1 A 3	RVEC23	RMB	3	\$B821	\$8304	₩025N	¥0LDL	CRUNCH BASIC LINE		
Ø39Ø		RVEC24	RMB	3	\$B7C2	<b>4035</b> 4			UNCRUNCH BASIC LINE		
Ø391	DINO	KVLUZ-	KIID	3	<b>4</b> D/02				ONCKONCH BASIC LINE		
	Ø1 A O	CIDCIN	RMB	8*5	CTDINC	DESCRIPTO	OD STACK				
Ø392		STRSTK				DESCRIPTO		D			
Ø393		CFNBUF	RMB	9		E FILE NA					
Ø394		CASBUF	RMB	256		E FILE DA					
Ø395		LINHDR	RMB	2		IPUT BUFFI					
Ø396		LINBUF	RMB	LBUFMX+1		.INE INPU	I BUFFER				
Ø397	Ø3D7	STRBUF	RMB	41	STRING	BUFFER					
Ø398											
Ø399	0400	VIDRAM	RMB	200	VIDEO D	ISPLAY A	REA				
0400											
0401		*START OF AD	DITIONA	L RAM VARIABLE STOR	AGE (DIS	K BASIC (	ONLY)				
0402	0600	DBUFØ	RMB	SECLEN	I/O BUF	FER #Ø					
0403	0700	DBUF1	RMB	SECLEN	I/O BUF	FER #1					
0404	0800	FATBLØ	RMB	FATLEN	FILE AL	LOCATION	TABLE -	DRIVE Ø			
0405	Ø84A	FATBL1	RMB	FATLEN	FILE AL	LOCATION	TABLE -	DRIVE 1			
0406	Ø894	FATBL2	RMB	FATLEN	FILE AL	LOCATION	TABLE -	DRIVE 2			
0407		FATBL3	RMB	FATLEN		LOCATION					
0408		FCBV1	RMB	16*2		IFFER VEC			SYSTEM)		
	Ø948	RNBFAD	RMB	2		F FREE RA					
	Ø94A	FCBADR	RMB	2		F FILE CO					
Ø411		DNAMBF	RMB	8		LE NAME E					
Ø412		DEXTBF	RMB	3		LE EXTENS		BUFFER			
Ø413		DFLTYP	RMB	1					OATA, 2=MACHINE		
Ø414		*				NGUAGE,					
Ø414	Ø958	DASCFL	RMB	1					NARY, \$FF=ASCII		
Ø416		DRUNFL	RMB	1					F BIT Ø=1, THEN CLOSE		
Ø417		*	2	-		ES BEFORI			2 1, 02002		
Ø418	MOE V	DEFDRV	RMB	1		DRIVE N		,			
Ø419		FCBACT	RMB	1		OF FCBS		CE 4 11151	II A CHUT DOUBLALL FORC		
Ø42Ø		DRESFL	RMB	1					N' & SHUT DOWN ALL FCBS		
Ø421		DLOADFL	RMB	1					IG A LOAD ERROR		
Ø422		DMRGFL	RMB	1		LAG: Ø=N					
Ø423		DUSRVC	RMB	20		SIC USR (	CUMMAND V	ECTURS			
Ø424				AREA FOR DIRECTORY	SEARCH						
0425		* EXISTING									
Ø426	0973	V973	RMB	1	SECTOR	NUMBER					
0427	0974	V974	RMB	2	RAM DIR	ECTORY I	MAGE ADDR	ESS			
Ø428	Ø976	V976	RMB	1	FIRST G	RANULE N	UMBER				
Ø429		* UNUSED F	ILE								
0430	Ø977	V977	RMB	1	SECTOR	NUMBER					
			RMB	2		RECTORY IN	MAGE ADDR	ESS			
	Ø978	V9/8									
	0978	V978	ICID								
Ø432				2	WRITF F	AT VALUE	: NUMRER	OF FRFF (	GRANULES WHICH MUST BE TAKEN		
Ø432 Ø433		WFATVL	RMB	2					GRANULES WHICH MUST BE TAKEN FAT TO DISK SEQUENCE		
Ø432	Ø97A			2	FROM TH		TRIGGER	A WRITE F	GRANULES WHICH MUST BE TAKEN FAT TO DISK SEQUENCE		

Ø436	Ø97E	DRØTRK	RMB	4	CURRENT TRACK NUMBER, DRIVES Ø,1,2,3
Ø437	Ø982	NMIFLG	RMB	1	NMI FLAG: Ø=DON'T VECTOR ∽Ø=YECTOR OUT
Ø438	Ø983	DNMIVC	RMB	2	NMI VECTOR: WHERE TO JUMP FOLLOWING AN NMI
Ø439		*			INTERRUPT IF THE NMI FLAG IS SET
	0985	RDYTMR	RMB	1	MOTOR TURN OFF TIMER
0441	0986	DRGRAM	RMB	1	RAM IMAGE OF DSKREG (\$FF40)
0442	Ø987	DVERFL	RMB	1	VERIFY FLAG: Ø=OFF, \$FF=ON
Ø443	Ø988	ATTCTR	RMB	1	READ/WRITE ATTEMPT COUNTER: NUMBER OF TIMES THE
0444		*			DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA
Ø445		*			BEFORE IT GIVES UP AND ISSUES AN ERROR.
Ø446					
Ø447	Ø989	DFLBUF	RMB	SECLEN	INITIALIZED TO SECLEN BY DISKBAS
Ø448					
Ø449		*RANDOM FIL	E RESER	VED AREA	
0450					
Ø451		*FILE CONTR	OL BLOCI	KS AND BUFFERS	
Ø452					
Ø453		*GRAPHIC PA	iE RESEI	RVED AREA	
Ø454		+04676 0000			
Ø455		*BASIC PROG	KAM		
Ø456		+VADIADIE C	TODACE	ADEA	
Ø457		*VARIABLE S	IUKAGE /	AKEA	
Ø458 Ø459		*ARRAY STOR	ACE ADE		
Ø459 Ø46Ø		ARRAT STUR	AGE AKE	Α.	
Ø461					
Ø462		* FREE MEMO	ov		
Ø463		TREE HENO	VI.		
Ø464					
Ø465		*STACK			
Ø466					
Ø467		*STRING SPA	CE		
Ø468					
Ø469		*USER PROGRA	AM RESEI	RVED AREA	
0470					
0471		*END OF RAM			
Ø472					
Ø473	8000	ORG	\$8000		
Ø474					
Ø475	8000		RMB	\$2000	EXTENDED BASIC ROM
	A000		RMB	\$2000	COLOR BASIC ROM
Ø477		ROMPAK	EQU	*	
	CØØØ	DOSBAS	RMB	\$2000	DISK BASIC ROM/ENHANCED BASIC INIT CODE
Ø479	FNNN		RMB	\$1F00	ENHANCED BASIC
0480					
0481		+ 1/0 4054			
Ø482 Ø483		* I/O AREA			
Ø484	EEAA	PIAØ	EQU	*	PERIPHERAL INTERFACE ADAPTER ONE
Ø485	רושש	FIAD	LQU		PERIFFIERAL INTERFACE ADAPTER ONE
Ø486	FFØØ	BITØ	KEYROA	ARD ROW 1 AND RIGHT	JOYSTICK SWITCH 1
Ø487		BIT1		ARD ROW 2 AND LEFT J	
Ø488		BIT2		ARD ROW 3 AND RIGHT	
Ø489		BIT3		ARD ROW 4 AND LEFT J	
0490		BIT4		ARD ROW 5	
0491		BIT5	KEYBOA	ARD ROW 6	
Ø492		BIT6	KEYBOA	ARD ROW 7	
Ø493		BIT7	JOTSTI	CK COMPARISON IINPU	Т
Ø494					
	FFØ1	BITØ		OL OF HSYNC (63.5ps)	
Ø496			INTERF		1 = IRQ* TO CPU ENABLED
Ø497		BIT1		OL OF INTERRUPT	Ø = FLAG SET ON FALLING EDGE OF HS
Ø498			POLARI		1 = FLAG SET ON RISING EDGE OF HS
Ø499		BIT2	NORMAL		Ø = CHANGES FFØØ TO DATA DIRECTION
0500		BIT3	SEL 1		LSB OF TWO ANALOG MUX SELECT LINES
Ø5Ø1		BIT4	ALWAYS		
Ø5Ø2		BIT5	ALWAYS		
Ø5Ø3 Ø5Ø4		BIT6 BIT7	NOT US	SED ONTAL SYNC INTERRUPT	ELAC
Ø5Ø5		DII/	пикти	MIME SING INTERKUPT	FLAG
	FFØ2	вітю	KEADU	ARD COLUMN 1	
Ø5Ø7	1104	BIT1		ARD COLUMN 1	
Ø5Ø8		BIT2		ARD COLUMN 3	
Ø5Ø9		BIT3		ARD COLUMN 4	
Ø51Ø		BIT4		ARD COLUMN 5	
Ø511		BIT5		ARD COLUMN 6	
Ø512		BIT6		ARD COLUMN 7 / RAM S	IZE OUTPUT
Ø513		BIT7		ARD COLUMN 8	•
Ø514				-	
	FFØ3	BITØ	CONTRO	DL OF VSYNC (16.667π	s) Ø = IRQ* TO CPU DISABLED
Ø516			INTERF	RUPT	1 = IRQ* TO CPU ENABLED
Ø517		BIT1	CONTRO	OL OF INTERRUPT	Ø = FLAG SET ON FALLING EDGE OF FS
Ø518			POLARI		1 = FLAG SET ON RISING EDGE OF FS
Ø519		BIT2	NORMAL		Ø = CHANGES FFØ2 TO DATA DIRECTION
Ø52Ø		BIT3	SEL 2		MSB OF TWO ANALOG MUX SELECT LINES
Ø521		BIT4	ALWAYS		
Ø522		BIT5	ALWAYS	5 1	

Ø523							
2020		BIT6	NOT US	ED			
Ø524		BIT7	FIELD	SYNC INTERRUPT FLAG			
Ø525							
Ø526	FFØ4		RMB	28	PIAØ IMA	AGES	
Ø527		DA					
Ø528		PIA1	EQU	*	PERTPHER	RAL INTERFACE ADAPTER TWO	
Ø529			-40				
Ø53Ø	FF2Ø	BITØ	CASSET	TE DATA INPUT			
Ø538	1120	BIT1		C DATA OUTPUT			
Ø532		BIT2		D/A LSB			
Ø533		BIT3	6 BIT				
Ø534		BIT4	6 BIT				
Ø535		BIT5	6 BIT				
Ø536		BIT6	6 BIT				
Ø537		BIT7	6 BIT	D/A MSB			
Ø538							
Ø539	FF21	BITØ	CONTRO	L OF CD		Ø = FIRQ* TO CPU DISABLE	)
0540			(RS-23	2C STATUS)		1 = FIRQ* TO CPU ENABLED	
Ø541		BIT1	CONTRO	L OF INTERRUPT		Ø = FLAG SET ON FALLING I	EDGE OF CD
Ø542			POLARI	TY		1 = FLAG SET ON RISING E	DGE OF CD
Ø543		BIT2	NORMAL	LY 1		Ø = CHANGES FF2Ø TO DATA	DIRECTION
Ø544		BIT3		TE MOTOR CONTROL		$\emptyset = OFF$ $1 = ON$	
Ø545		BIT4	ALWAYS			2 0	
Ø546		BIT5	ALWAYS				
Ø547			NOT US				
		BIT6					
Ø548		BIT7	CD INI	ERRUPT FLAG			
Ø549							
Ø55Ø	FFZZ	BITØ		C DATA INPUT			
Ø551		BIT1		BIT SOUND OUTPUT			
Ø552		BIT2		ZE INPUT			
Ø553		BIT3		NITOR SENSING INPUT		CSS	
Ø554		BIT4		NTROL OUTPUT		GMØ & UPPER/LOWER CASE*	
Ø555		BIT5	VDG CO	NTROL OUTPUT		GM1 & INVERT	
Ø556		BIT6	VDG CO	NTROL OUTPUT		GM2	
Ø557		BIT7	VDG CO	NTROL OUTPUT		A*/G	
Ø558							
Ø559	FF23	BITØ	CONTRO	L OF CARTRIDGE		Ø = FIRQ* TO CPU DISABLE	)
Ø56Ø			INTERR			1 = FIRQ* TO CPU ENABLED	
Ø561		BIT1	CONTRO	L OF INTERRUPT		Ø = FLAG SET ON FALLING	FDGE OF CART*
Ø562		D111	POLARI			1 = FLAG SET ON RISING E	
Ø563		BIT2	NORMAL			Ø = CHANGES FF22 TO DATA	
						W - CHANGES FFZZ TO DATA	DIRECTION
Ø564		BIT3	SOUND				
Ø565		BIT4	ALWAYS				
Ø566		BIT5	ALWAYS				
Ø567		BIT6	NOT US				
Ø568		BIT7	CARTRI	DGE INTERRUPT FLAG			
Ø569							
Ø57Ø	FF24		RMB	28	PIA1 IMA	AGES	
Ø57Ø Ø571		PIA2	RMB	28	PIA1 IMA	AGES	
	FF4Ø	PIA2 DSKREG	RMB RMB	28		AGES ITROL REGISTER	
Ø571	FF4Ø						
Ø571 Ø572	FF40 FF40		RMB				
Ø571 Ø572 Ø573 Ø574	FF40 FF40	DSKREG BITØ	RMB DRIVE	1 SELECT Ø			
0571 0572 0573 0574 0575	FF40 FF40	DSKREG BITØ BIT1	RMB DRIVE DRIVE	1 SELECT Ø SELECT 1			
0571 0572 0573 0574 0575 0576	FF40 FF40	DSKREG BITØ BIT1 BIT2	RMB DRIVE DRIVE DRIVE	1 SELECT Ø SELECT 1 SELECT 2		ITROL REGISTER	1 = MOTORS ON
Ø571 Ø572 Ø573 Ø574 Ø575 Ø576	FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3	RMB  DRIVE DRIVE DRIVE DRIVE	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE		NTROL REGISTER  Ø = MOTORS OFF	1 = MOTORS ON 1 = PRECOMP
Ø571 Ø572 Ø573 Ø574 Ø575 Ø576 Ø577	FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3 BIT4	RMB  DRIVE DRIVE DRIVE DRIVE WRITE	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION		<pre>### ITROL REGISTER  ### ### ### ### ### ### #### #### ##</pre>	1 = PRECOMP
Ø571 Ø572 Ø573 Ø574 Ø575 Ø576 Ø577 Ø578	FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG		NTROL REGISTER  Ø = MOTORS OFF	
Ø571 Ø572 Ø573 Ø574 Ø575 Ø576 Ø577 Ø578 Ø579	FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3		Ø = MOTORS OFF Ø = NO PRECOMP Ø = SINGLE	1 = PRECOMP 1 = DOUBLE
Ø571 Ø572 Ø573 Ø574 Ø575 Ø576 Ø577 Ø578 Ø579 Ø58Ø Ø581	FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3		<pre>### ITROL REGISTER  ### ### ### ### ### ### #### #### ##</pre>	1 = PRECOMP
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581	FF40 FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG	DISK CON	WITROL REGISTER  Ø = MOTORS OFF Ø = NO PRECOMP Ø = SINGLE Ø = DISABLED	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583	FF40 FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3		WITROL REGISTER  Ø = MOTORS OFF Ø = NO PRECOMP Ø = SINGLE Ø = DISABLED	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0578 0580 0581 0582 0583	FF40 FF40 FF40	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7	RMB  DRIVE DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG	DISK CON	WITROL REGISTER  Ø = MOTORS OFF Ø = NO PRECOMP Ø = SINGLE Ø = DISABLED	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI	DISK CON DSKREG I STERS	WAGES	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0585	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7	RMB  DRIVE DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG	DISK CON DSKREG I STERS	WITROL REGISTER  Ø = MOTORS OFF Ø = NO PRECOMP Ø = SINGLE Ø = DISABLED	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  K CONTR	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI-	DISK CON DSKREG I STERS	### MOTORS OFF  ### Ø = MOTORS OFF  ### Ø = NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ### MAGES  **COMMAND REGISTER	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7	RMB  DRIVE DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  K CONTR	1 SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI- 1 COMMAND	DISK CON DSKREG I STERS	W = MOTORS OFF Ø = NO PRECOMP Ø = SINGLE Ø = DISABLED MAGES COMMAND REGISTER CODE	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0580 0581 0582 0583 0584 0585 0586 0587	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB K CONTR	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI: 1 COMMAND RESTORE	DISK CON DSKREG I	### WOTORS OFF  ### ### ### ### ### #### ##########	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0580 0581 0582 0583 0584 0585 0586 0587 0588	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  K CONTF	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK	DISK CON DSKREG I	### MOTORS OFF  ### ### ### ### ### #### #### ####	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0584 0585 0586 0587 0588	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE DRIVE DRIVE HALT F RMB  TYPE I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI- 1 COMMAND RESTORE SEEK STEP	DISK CON DSKREG I	### MOTORS OFF  ### Ø = MOTORS OFF  ### Ø = NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ### MAGES  ### CODE  ### ### ### ### ### ### ### ### ### #	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0588 0582 0583 0584 0585 0586 0587 0588	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE URIVE WRITE DENSIT DRIVE HALT F RMB  K CONTF RMB  TYPE I I I I	SELECT Ø SELECT 1 SELECT 2 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGIL 1 COMMAND RESTORE SEEK STEP STEP	DISK CON DSKREG I	### WOTORS OFF  ### Ø = MOTORS OFF  ### Ø = NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  #### BAGES  #### CODE  ### ### ### ### ### ###  ### CODE  ### ### ### ### ### ### ### ### ### #	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586 0587 0588 0590 0591 0592	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  K CONTR RMB  TYPE I I I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT	DISK CON DSKREG I	### MOTORS OFF  ### Ø = MOTORS OFF  ### Ø = NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ###################################	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0585 0586 0587 0588 0589 0591 0592	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE DRIVE DENSIT DRIVE HALT F  RMB  TYPE I I I I I I II	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI- 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR	DISK CON DSKREG I	### MOTORS OFF  ### ### ### ### ### ### #### ####	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586 0587 0588 0599 0590 0591 0592	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE URIVE WRITE DENSIT DRIVE HALT F RMB  TYPE I I I I I I III	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI- 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR	DISK CON DSKREG I	### WOTORS OFF  ### ### ### ### ### ### #### ####	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586 0587 0598 0599 0591 0592 0593 0594 0595	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  TYPE I I I I I I I III III	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS	DISK CON DSKREG I	### WOTORS OFF  ### Ø = MOTORS OFF  ### ### Ø = NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ###################################	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0585 0586 0587 0589 0591 0592 0593 0594 0595	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK	DISK CON DSKREG I	### MOTORS OFF  ### Ø = MOTORS OFF  ### NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ###################################	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586 0587 0598 0599 0591 0592 0593 0594 0595	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  TYPE I I I I I I I III III	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS	DISK CON DSKREG I	### WOTORS OFF  ### Ø = MOTORS OFF  ### ### Ø = NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ###################################	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0585 0586 0587 0589 0591 0592 0593 0594 0595	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK	DISK CON DSKREG I	### MOTORS OFF  ### Ø = MOTORS OFF  ### NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ###################################	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0580 0581 0582 0583 0584 0585 0586 0587 0591 0591 0592 0593 0594 0595 0597 0598	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F  RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI- 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK	DISK CON DSKREG I	### MOTORS OFF  ### MOTORS OFF	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0588 0582 0583 0584 0585 0586 0587 0591 0592 0593 0594 0595 0596 0597	FF40 FF40 FF41	DSKREG  BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F  RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI- 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK	DSKREG I STERS STATUS/C	### MOTORS OFF  ### MOTORS OFF	1 = PRECOMP 1 = DOUBLE
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586 0587 0588 0599 0591 0592 0593 0594 0595 0597 0598	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  K CONTF RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI: 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT	DSKREG I STERS STATUS/C	### WOTORS OFF  ### ### ### ### ### ### ### ### #### ####	1 = PRECOMP 1 = DOUBLE 1 = ENABLED
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0585 0586 0587 0591 0591 0592 0593 0591 0595 0596 0597 0598 0599 0690 0601	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F  RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT TYPE I	DSKREG I STERS STATUS/C	### WOTORS OFF  ### ### ### ### ### ### ### ### #### ####	1 = PRECOMP 1 = DOUBLE  1 = ENABLED  WRITE SECTOR/TRACK
0571 0572 0573 0574 0575 0576 0577 0578 0579 0588 0582 0588 0585 0586 0587 0599 0591 0592 0593 0594 0595 0597 0598 0597 0598 0600 0601 0602	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT TYPE I BUSY INDEX	DISK CON  DSKREG I  STERS STATUS/C	### MOTORS OFF  ### Ø = MOTORS OFF  ### WO PRECOMP  ### Ø = DISABLED  ### BAGES  #### BAGES  ##### BAGES  ##### BAGES  ##### BAGES  ###################################	1 = PRECOMP 1 = DOUBLE 1 = ENABLED WRITE SECTOR/TRACK BUSY DRQ
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586 0587 0591 0592 0593 0594 0595 0596 0597 0598	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT TYPE I BUSY INDEX TRACK Ø	DSKREG I STERS STATUS/C  READ ADD BUSY DRQ LOST DAT	### MOTORS OFF ### #### MOTORS OFF ### MOTORS OFF ### MO	1 = PRECOMP 1 = DOUBLE 1 = ENABLED WRITE SECTOR/TRACK BUSY DRQ LOST DATA
0571 0572 0573 0574 0575 0576 0577 0578 0579 0580 0581 0582 0583 0584 0585 0586 0587 0591 0591 0593 0594 0595 0595 0596 0597 0598 0599 0600 0601 0602 0603	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE DRIVE HALT F  RMB  K CONTR RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR WRITE SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT  TYPE I BUSY INDEX TRACK Ø CRC ERROR	DSKREG I STERS STATUS/C	### MOTORS OFF ### MO	1 = PRECOMP 1 = DOUBLE  1 = ENABLED  WRITE SECTOR/TRACK BUSY DRQ LOST DATA CRC ERROR (EXCEPT TRACK)
0571 0572 0573 0574 0575 0576 0577 0578 0579 0588 0582 0583 0584 0585 0586 0587 0591 0592 0593 0594 0595 0596 0597 0598 0599 0600 0601 0602 0603 0604	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE URITE DRIVE HALT F RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI. 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT  TYPE I BUSY INDEX TRACK Ø CRC ERROR SEEK ERROR	DISK CON  DSKREG I  STERS STATUS/C  READ ADD BUSY DRQ LOST DAT CRC ERRC RNF (EXC	### MOTORS OFF  ### Ø = MOTORS OFF  ### ### Ø = NO PRECOMP  ### Ø = SINGLE  ### Ø = DISABLED  ### MAGES  ### M	1 = PRECOMP 1 = DOUBLE 1 = ENABLED WRITE SECTOR/TRACK BUSY DRQ LOST DATA CRC ERROR (EXCEPT TRACK) RNF (EXCEPT TRACK)
0571 0572 0573 0574 0575 0576 0577 0578 0579 0581 0582 0583 0584 0585 0586 0587 0599 0591 0592 0593 0594 0595 0596 0697 0608 0601 0603 0604 0605 0606	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE WRITE DENSIT DRIVE HALT F RMB  K CONTF RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG 7 ROLLER INTERNAL REGI 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT TYPE I BUSY INDEX TRACK Ø CRC ERROR SEEK ERROR HEAD LOADED	DISK CON  DSKREG I  STERS STATUS/C  READ ADD BUSY DRQ LOST DAT CRC ERRC RNF (EXC	### MOTORS OFF ### MO	1 = PRECOMP 1 = DOUBLE  1 = ENABLED  WRITE SECTOR/TRACK BUSY DRQ LOST DATA CRC ERROR (EXCEPT TRACK) WRITE FAULT
0571 0572 0573 0574 0575 0576 0577 0578 0579 0588 0582 0583 0584 0585 0586 0587 0591 0592 0593 0594 0595 0596 0597 0598 0599 0600 0601 0602 0603 0604	FF40 FF40 FF41	DSKREG BITØ BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7  * FLOPPY DIS FDCREG COMMANDS	RMB  DRIVE DRIVE DRIVE DRIVE URITE DRIVE HALT F RMB  TYPE I I I I I I I I I I I I I I I I I I I	SELECT Ø SELECT 1 SELECT 1 SELECT 2 MOTOR ENABLE PRECOMPENSATION Y FLAG SELECT 3 LAG  7 ROLLER INTERNAL REGI. 1 COMMAND RESTORE SEEK STEP STEP IN STEP OUT READ SECTOR WRITE SECTOR READ ADDRESS READ TRACK WRITE TRACK FORCE INTERRUPT  TYPE I BUSY INDEX TRACK Ø CRC ERROR SEEK ERROR	DISK CON  DSKREG I  STERS STATUS/C  READ ADD BUSY DRQ LOST DAT CRC ERRC RNF (EXC	### MOTORS OFF ### MOTORS OFF ### NO PRECOMP ### SINGLE ### DISABLED ####################################	1 = PRECOMP 1 = DOUBLE 1 = ENABLED WRITE SECTOR/TRACK BUSY DRQ LOST DATA CRC ERROR (EXCEPT TRACK) RNF (EXCEPT TRACK)

0610					
Ø611	FF49		RMB	1	TRACK REGISTER
Ø612	FF4A		RMB	1	SECTOR REGISTER
Ø613	FF4B		RMB	1	DATA REGISTER
Ø614	FF4C		RMB	4	FDCREG IMAGES
Ø615					
Ø616	FF5Ø		RMB	16	UNUSED SPACE
Ø617	FF6Ø		RMB	1	X COORDINATE FOR X-PAD
	FF61		RMB	1	Y COORDINATE FOR X-PAD
Ø619	FF62		RMB	1	STATUS REGISTER FOR X-PAD
Ø62Ø			RMB	5	UNUSED
Ø621	FF03	* RS-232 PR0			UNUSED
Ø622	FF68	* K3-232 PKU	RMB	1	READ/WRITE DATA REGISTER
				=	
	FF69		RMB	1	STATUS REGISTER
Ø624	FF6A		RMB	1	COMMAND REGISTER
Ø625	FF6B		RMB	1	CONTROL REGISTER
Ø626	FF6C		RMB	4	
Ø627	FF7Ø		RMB	13	
Ø628	FF7D		RMB	1	SOUND/SPEECH CARTRIDGE RESET
Ø629	FF7E		RMB	1	SOUND/SPEECH CARTRIDGE READ/WRITE
Ø63Ø	FF7F		RMB	1	MULTI-PAK PROGRAMMING REGISTER
Ø631					
Ø632	FF8Ø		RMB	64	RESERVED FOR FUTURE EXPANSION
Ø633					
Ø634					
Ø635	FFCØ	SAMREG	EQU	*	SAM CONTROL REGISTERS
Ø636					
Ø637	FFCØ	VØCLR	RMB	1	CLEAR COCO GRAPHICS MODE VØ
Ø638	FFC1	VØSET	RMB	1	SET COCO GRAPHICS MODE VØ
Ø639	FFC2	V1CLR	RMB	1	CLEAR COCO GRAPHICS MODE V1
Ø64Ø	FFC3	V1SET	RMB	1	SET COCO GRAPHICS MODE V1
0641	FFC4	V2CLR	RMB	1	CLEAR COCO GRAPHICS MODE V2
	FFC5	V2SET	RMB	1	SET COCO GRAPHICS MODE V2
Ø643	FFC6	FØCLR	RMB	1	CLEAR COCO GRAPHICS OFFSET FØ
Ø644	FFC7	FØSET	RMR	1	SET COCO GRAPHICS OFFSET FØ
Ø645		F1CLR	RMB	1	CLEAR COCO GRAPHICS OFFSET F1
	FFC9	F1SET	RMB	1	SET COCO GRAPHICS OFFSET F1
Ø647	FFCA	F2CLR	RMB	1	CLEAR COCO GRAPHICS OFFSET F2
	FFCB	F2SET	RMB	1	SET COCO GRAPHICS OFFSET F2
Ø649	FFCC	F3CLR	RMB	1	CLEAR COCO GRAPHICS OFFSET F3
Ø65Ø	FFCD	F3SET	RMB	1	SET COCO GRAPHICS OFFSET F3
				-	
Ø651 Ø652	FFCE	F4CLR	RMB	1	CLEAR COCO GRAPHICS OFFSET F4
		F4SET	RMB	=	SET COCO GRAPHICS OFFSET F4
Ø653	FFDØ	F5CLR	RMB	1	CLEAR COCO GRAPHICS OFFSET F5
Ø654	FFD1	F5SET	RMB	1	SET COCO GRAPHICS OFFSET F5
Ø655	FFD2	F6CLR	RMB	1	CLEAR COCO GRAPHICS OFFSET F6
Ø656	FFD3	F6SET	RMB	1	SET COCO GRAPHICS OFFSET F6
Ø657	FFD4		RMB	4	RESERVED
Ø658	FFD8	R1CLR	RMB	1	CLEAR CPU RATE, (Ø.89 MHz)
Ø659	FFD9	R1SET	RMB	1	SET CPU RATE, (1.78 MHz)
Ø66Ø	FFDA		RMB	4	RESERVED
Ø661	FFDE	ROMCLR	RMB	1	ROM DISABLED
Ø662	FFDF	ROMSET	RMB	1	ROM ENABLED
Ø663					
Ø664	FFEØ		RMB	18	RESERVED FOR FUTURE MPU ENHANCEMENTS
Ø665		*	INTER	RRUPT VECTORS	
Ø666	FFF2	SWI3	RMB	2	
Ø667	FFF4	SWI2	RMB	2	
Ø668	FFF6	FIRQ	RMB	2	
Ø669	FFF8	IRQ	RMB	2	
Ø67Ø	FFFA	SWI	RMB	2	
	IIIA				
Ø671	FFFC	NMI	RMB	2	
			RMB RMB	2 2	

0001	AØØØ				ORG	\$AØØØ	
0002	AØØØ	A1 CB		POLCAT	FDB	KEYIN	GET A KEYSTROKE
0003	AØØ2	A2 82		CHROUT	FDB	PUTCHR	OUTPUT A CHARACTER
0004	AØØ4	A7 7C		CSRDON	FDB	CASON	TURN ON CASSETTE MOTOR, START READING
0005	AØØ6	A7 ØB		BLKIN	FDB	GETBLK	READ A BLOCK FROM CASSETTE
0006	AØØ8	A7 F4 A9 DE		BLKOUT JOYIN	FDB	SNDBLK	WRITE A BLOCK TO CASSETTE READ JOYSTICKS
0007 0008	AØØA AØØC	A9 DE A7 D8		WRTLDR	FDB FDB	GETJOY WRLDR	TURN ON MOTOR AND WRITE \$55 S TO CASSETTE
0009	ADDC	A7 D0		*	100	WKLDK	TORN ON HOTOR AND WRITE \$33 3 TO CASSETTE
0010	AØØE	10 CE 03	3 D7	LAØØE	LDS	#LINBUF+LBUFMX+1	SET STACK TO TOP OF LINE INPUT BUFFER
0011	AØ12	86 37			LDA	#\$37	*
0012	AØ14	B7 FF 23	3		STA	PIA1+3	* ENABLE 63.5 MICROSECOND INTERRUPT
0013	AØ17	96 71			LDA	RSTFLG	GET WARM START FLAG
0014	AØ19	81 55			CMPA	#\$55	IS IT A WARM START?
0015	AØ1B	26 57			BNE	BACDST	NO - DØ A COLD START
0016	AØ1D	9E 72			LDX	RSTVEC	WARM START VECTOR
0017	AØ1F	A6 84			LDA	, Х	GET FIRST BYTE OF WARM START ADDR
ØØ18 ØØ19	AØ21 AØ23	81 12 26 4F			CMPA BNE	#\$12 BACDST	IS IT NOP? NO - DO A COLD START
0019 0020	AØ25	6E 84			JMP	,X	YES, GØ THERE
0020	MULJ	0L 04			UHI	, ^	its, de intre
0022	AØ27	31 8C E4	4	RESVEC	LEAY	LAØØE,PC	POINT Y TO WARM START CHECK CODE
0023	AØ2A	8E FF 29		LAØ2A	LDX	#PIA1	POINT X TO PIA1
0024	AØ2D	6F 1D			CLR	-3,X	CLEAR PIAØ CONTROL REGISTER A
0025	AØ2F	6F 1F			CLR	-1,X	CLEAR PIAØ CONTROL REGISTER B
0026	AØ31	6F 1C			CLR	-4,X	SET PIAØ SIDE A TO INPUT
0027	AØ33	CC FF 34	4		LDD	#\$FF34	*
0028	AØ36	A7 1E			STA	-2,X	* SET PIAØ SIDE B TO OUTPUT
0029	AØ38	E7 1D			STB	-3,X	* ENABLE PIAØ PERIPHERAL REGISTERS, DISABLE PIAØ
0030	AØ3A	E7 1F			STB	-1,X	* MPU INTERRUPTS, SET CA2, CA1 TO OUTPUTS
0031	AØ3C	6F Ø1			CLR	1,X	CLEAR CONTROL REGISTER A ON PIA1
ØØ32 ØØ33	AØ3E AØ4Ø	6F Ø3 4A			CLR DECA	3,X	CLEAR CONTROL REGISTER B ON PIA1 A REG NOW HAS \$FE
ØØ34	AØ41	A7 84			STA	, Х	BITS 1-7 ARE OUTPUTS, BIT Ø IS INPUT ON PIA1 SIDE A
0035	AØ43	86 F8			LDA	#\$F8	=
0036	AØ45	A7 Ø2			STA	2,X	= BITS Ø-2 ARE INPUTS, BITS 3-7 ARE OUTPUTS ON B SIDE
0037	AØ47	E7 Ø1			STB	1,X	* ENABLE PERIPHERAL REGISTERS, DISABLE PIA1 MPU
0038	AØ49	E7 Ø3			STB	3,X	* INTERRUPTS AND SET CA2, CB2 AS OUTPUTS
0039	AØ4B	6F Ø2			CLR	2,X	SET 6847 MODE TO ALPHA-NUMERIC
0040	AØ4D	C6 Ø2			LDB	#\$02	*
0041	AØ4F	E7 84			STB	, χ	* MAKE RS232 OUTPUT MARKING
0042	AØ51	CE FF C	Ø		LDU	#SAMREG	SAM CONTROL REGISTER ADDR
0043	AØ54	C6 10			LDB	#16	16 SAM CONTROL REGISTER BITS
0044	AØ56	A7 C1		LAØ56	STA	, U++	ZERO OUT SAM CONTROL REGISTER BIT
0045	AØ58	5A			DECB	LAGEC	* DECREMENT COUNTER AND
ØØ46 ØØ47	AØ59 AØ5B	26 FB B7 FF C9	n		BNE	LAØ56	* BRANCH IF NOT DONE
0047 0048	AØ5E	1F 9B	9		STA TFR	SAMREG+9 B,DP	SET DISPLAY PAGE AT \$400 SET DIRECT PAGE TO ZERO
ØØ49	AØ6Ø	C6 Ø4			LDB	#\$04	USE AS A MASK TO CHECK RAMSZ INPUT
0050	AØ62	A7 1E			STA	-2,X	SET RAMSZ STROBE HIGH
0051	AØ64	E5 Ø2			BITB		CHECK RAMSZ INPUT
0052	AØ66	27 ØA			BEQ	LAØ72	BRANCH IF JUMPER SET FOR 4K RAMS
0053	AØ68	6F 1E			CLR	-2,X	SET RAMSZ STROBE LOW
0054	AØ6A	E5 Ø2			BITB	2,X	CHECK RAMSZ INPUT
0055	AØ6C	27 Ø2			BEQ	LAØ7Ø	BRANCH IF JUMPER SET FOR 64K RAMS
0056	AØ6E	33 5E			LEAU	-2,U	ADJUST POINTER TO SET SAM FOR 16K RAMS
0057	AØ7Ø	A7 5D		LAØ7Ø	STA	-3,U	PROGRAM SAM FOR 16K OR 64K RAMS
0058	AØ72	6E A4		LAØ72	JMP	, Ү	GO DO A WARM OR COLD START
ØØ59	1074	05 04 0		* COLD ST			DOINT V TO CLEAD 1CT 1/ OF DAM
0060 0061	AØ74 AØ77	8E Ø4 Ø: 6F 83	1	BACDST LAØ77	LDX CLR	#VIDRAM+1	POINT X TO CLEAR 1ST 1K OF RAM MOVE POINTER DOWN TWO-CLEAR BYTE
	AØ79	3Ø Ø1		LAW//		, X 1 Y	ADVANCE POINTER ONE
0062 0063	AØ79 AØ7B	26 FA			LEAX BNE	1,X LAØ77	KEEP GOING IF NOT AT BOTTOM OF PAGE Ø
0064	AØ7D	BD A9 28	В		JSR	LA928	CLEAR SCREEN
0065	AØ8Ø	6F 8Ø	-		CLR	, X+	CLEAR 1ST BYTE OF BASIC PROGRAM
0066	AØ82	9F 19			STX	TXTTAB	BEGINNING OF BASIC PROGRAM
0067	AØ84	A6 Ø2		LAØ84	LDA	2,X	LOOK FOR END OF MEMORY
0068	AØ86	43			COMA		* COMPLEMENT IT AND PUT IT BACK
0069	AØ87	A7 Ø2			STA	2,X	* INTO SYSTEM MEMORY
0070	AØ89	A1 Ø2			CMPA	2,X	IS IT RAM?
0071	AØ8B	26 Ø6			BNE	LAØ93	BRANCH IF NOT (ROM, BAD RAM OR NO RAM)
0072	AØ8D	30 01			LEAX	1,X	MOVE POINTER UP ONE
0073	AØ8F	63 Ø1			COM	1,X	RE-COMPLEMENT TO RESTORE BYTE
0074	AØ91	20 F1		1.4000	BRA	LAØ84	KEEP LOOKING FOR END OF RAM
0075	AØ93	9F 74		LAØ93	STX	TOPRAM	SAVE ABSOLUTE TOP OF RAM
ØØ76	AØ95	9F 27 9F 23			STX	MEMSIZ	SAVE TOP OF STRING SPACE
ØØ77 ØØ78	AØ97 AØ99	9F 23 3Ø 89 FI	F 38		STX	STRTAB -200,X	SAVE START OF STRING VARIABLES CLEAR 200 - DEFAULT STRING SPACE TO 200 BYTES
ØØ79	AØ9D	9F 21	. 50		STX	FRETOP	SAVE START OF STRING SPACE TO 200 BITES
0080	AØ9F	1F 14			TFR	X,S	PUT STACK THERE
0081	AØA1	8E A1 ØI	D		LDX	#LA1ØD	POINT X TO ROM SOURCE DATA
0082	AØA4	CE ØØ 81			LDU	#CMPMID	POINT U TO RAM DESTINATION
ØØ83	AØA7	C6 1C			LDB	#28	MOVE 28 BYTES
0084	AØA9	BD A5 9/	A		JSR	LA59A	MOVE 28 BYTES FROM ROM TO RAM
0085	AØAC	CE Ø1 Ø	C		LDU	#IRQVEC	POINT U TO NEXT RAM DESTINATION
0086	AØAF	C6 1E			LDB	#30	MOVE 30 MORE BYTES
0087	AØB1	BD A5 9/	A		JSR	LA59A	MOVE 3Ø BYTES FROM ROM TO RAM
ØØ88	AØB4	AE 14		LAGDE	LDX	-12,X	POINT X TO SYNTAX ERROR ADDRESS
ØØ89	AØB6	AF 43		LAØB6	STX	3,U	* SET EXBAS COMMAND INTERPRETATION

0090	AMRO	AF 48					STX	8,U	* HANDLERS TO SYNTAX ERROR
ØØ91		8E Ø1					LDX	#RVECØ	POINT X TO START OF RAM VECTORS
0092	AØBD	CC 39					LDD	#\$394B	SET UP TO SAVE 75 RTS
0093	AØCØ	A7 8Ø				LAØCØ	STA	, X+	FILL THE RAM VECTORS WITH RTS
0094	AØC2						DECB		* DECREMENT COUNTER AND
ØØ95		26 FB B7 Ø2					BNE STA	LAØCØ	* BRANCH IF NOT DONE
ØØ96 ØØ97		BD AD						LINHDR-1 LAD19	PUT RTS IN LINHDR-1 GØ DO A NEW
ØØ98		8E 45					LDX	#\$4558	ASCII EX (FIRST TWO LETTERS OF EXTENDED )
0099		BC 8Ø						EXBAS	SEE IF EXTENDED ROM IS THERE
0100	AØD1	10 27	DF	2 D			LBEQ	EXBAS+2	IF IT IS, BRANCH TO IT
0101		1C AF					ANDCC		ENABLE IRQ, FIRQ
0102		8E A1					LDX	#LA147-1	POINT X TO COLOR BASIC COPYRIGHT MESSAGE
Ø1Ø3 Ø1Ø4		BD B9 8E AØ					JSR LDX	LB99C #BAWMST	PRINT COLOR BASIC WARM START ADDRESS
0105		9F 72						RSTVEC	SAVE IT
0106		86 55					LDA	#\$55	WARM START FLAG
0107	AØE4	97 71					STA	RSTFLG	SAVE IT
0108		2Ø ØB					BRA	LAØF3	GO TO BASIC S MAIN LOOP
Ø1Ø9	AØE8					BAWMST	NOP	DEVANIM	NOP REQ D FOR WARM START
Ø11Ø Ø111		ØF 6F BD AD					CLR JSR	DEVNUM LAD33	SET DEVICE NUMBER TO SCREEN DO PART OF A NEW
Ø112		1C AF					ANDCC		ENABLE IRQ,FIRQ
Ø113		BD A9					JSR	LA928	CLEAR SCREEN
0114	AØF3	7E AC	73			LAØF3	JMP	LAC73	GO TO MAIN LOOP OF BASIC
Ø115						*		OUTTNE	
Ø116 Ø117	Vale	7D FF	23			* FIRQ S BFRQSV	TST	PIA1+3	CARTRIDGE INTERRUPT?
Ø118		2B Ø1				DINQUI	BMI	LAØFC	YES
Ø119	AØFB	3B					RTI		
0120	AØFC					LAØFC		LA7D1	DELAY FOR A WHILE
0121		BD A7						LA7D1	KEEP DELAYING
Ø122 Ø123		31 8C 7E AØ				LA102	JMP	<la108,pc LA02A</la108,pc 	Y = ROM-PAK START UP VECTOR GO DO INITIALIZATION
Ø124		ØF 71				LA108	CLR	RSTFLG	CLEAR WARM START FLAG
0125	A1ØA	7E CØ	ØØ				JMP	ROMPAK	JUMP TO EXTERNAL ROM PACK
Ø126						*			
Ø127 Ø128	A1ØD	10				* THESE LA1ØD	BYTES AF FCB	RE MOVED TO ADDRESSES \$8F -	- \$AA THE DIRECT PAGE MID BAND PARTITION OF 1200/2400 HERTZ PERIOD
Ø128		18				LAIMD	FCB	18 24	UPPER LIMIT OF 1200 HERTZ PERIOD
0130	A1ØF						FCB	10	UPPER LIMIT OF 2400 HERTZ PERIOD
0131	A11Ø	ØØ 8Ø					FDB	128	NUMBER OF 55 S TO CASSETTE LEADER
Ø132	A112						FCB	11	CURSOR BLINK DELAY
Ø133		00 58					FDB	88	CONSTANT FOR 600 BAUD VER 1.2 & UP
Ø134 Ø135	A115	00 01 10					FDB FCB	1 16	PRINTER CARRIAGE RETURN DELAY TAB FIELD WIDTH
Ø136	A118						FCB	112	LAST TAB ZONE
Ø137	A119						FCB	132	PRINTER WIDTH
Ø138	A11A						FCB	Ø	LINE PRINTER POSITION
Ø139	A11B	B4 4A				+ 1785 7	FDB	LB44A	ARGUMENT OF EXEC COMMAND - SET TO FC ERROR
Ø14Ø Ø141	A11D	ØC A7				* LINE I	INC	CHARAD+1	
0142	MIID						BNE	LA123	
	A11F							CHARAD	
Ø143		ØC A6					INC	UIIAIAAD	
Ø144	A121 A123	ØC A6 B6 ØØ	ØØ			LA123	LDA	>0000	
Ø144 Ø145	A121 A123	ØC A6	ØØ						
Ø144 Ø145 Ø146	A121 A123	ØC A6 B6 ØØ	ØØ			*	LDA JMP	>0000 BROMHK	-\$129
0144 0145 0146 0147	A121 A123	ØC A6 B6 ØØ 7E AA	ØØ 1A			*	LDA JMP	>0000	-\$129 IRQ SERVICE
Ø144 Ø145 Ø146 Ø147 Ø148 Ø149 >	A121 A123 A126 A129 A120	ØC A6 B6 ØØ 7E AA 7E A9 7E AØ	ØØ 1A B3 F6			*	LDA JMP BYTES AF JMP JMP	>0000 BROMHK RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV	IRQ SERVICE FIRQ SERVICE
Ø144 Ø145 Ø146 Ø147 Ø148 Ø149 >	A121 A123 A126 A129 A12C A12F	ØC A6 B6 ØØ 7E AA 7E A9 7E AØ 7E B4	00 1A B3 F6 4A			*	LDA JMP BYTES AF JMP JMP JMP	>0000 BROMHK RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR)
0144 0145 0146 0147 0148 0149 > 0150 0151	A121 A123 A126 A129 A12C A12F A132	ØC A6 B6 ØØ 7E AA 7E A9 7E AØ 7E B4 8Ø 4F	ØØ 1A B3 F6 4A			*	LDA JMP BYTES AF JMP JMP JMP FCB	>0000 BROMHK RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED
0144 0145 0146 0147 0148 0149 > 0150 0151 0152	A121 A123 A126 A129 A12C A12F A132 A133	ØC A6 B6 ØØ 7E AA 7E A9 7E AØ 7E B4 8Ø 4F 4F C7	ØØ 1A B3 F6 4A			*	LDA JMP BYTES AF JMP JMP JMP	>0000 BROMHK RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR)
0144 0145 0146 0147 0148 0149 > 0150 0151	A121 A123 A126 A129 A12C A12F A132 A133 A135	ØC A6 B6 ØØ 7E AA 7E A9 7E AØ 7E B4 8Ø 4F 4F C7 52 59	ØØ 1A B3 F6 4A			*	LDA JMP BYTES AF JMP JMP FCB FDB FCB	>0000 BROMHK RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA
0144 0145 0146 0147 0148 0149 > 0150 0151 0152 0153 0154 0155	A121 A123 A126 A129 A12C A12F A132 A133 A135 A137 A138	7E A9 7E A9 7E B4 80 4F 4F C7 52 59 FF 04 5E	ØØ 1A B3 F6 4A			*	LDA JMP BYTES AF JMP JMP JMP FCB FDB FCB FCB FCB	>0000 BROMHK RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY
0144 0145 0146 0147 0148 0149 > 0150 0151 0152 0153 0154 0155	A121 A123 A126 A129 A12C A12F A132 A133 A135 A137 A138	7E A9 7E A9 7E B4 80 4F 4F C7 52 59 FF 04 5E	ØØ 1A B3 F6 4A			* * THESE	LDA JMP BYTES AF JMP JMP FCB FCB FCB FCB FCB FCB FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) **RANDON SEED **RANDON SEED OF MANTISSA ** 811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR)
0144 0145 0146 0147 0148 0149 > 0150 0151 0152 0153 0154 0155 0156 0157	A121 A123 A126 A129 A12C A12F A132 A133 A135 A137 A138 A13A	7E A9 7E A9 7E B4 80 4F 4F C7 52 59 FF 04 5E 7E B2	ØØ 1A B3 F6 4A			* * THESE  * BASIC	LDA JMP BYTES AF JMP JMP FCB FCB FCB FCB FCB FCB FCB FCB FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRGSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) **AANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE
0144 0145 0146 0147 0148 0149 > 0150 0151 0152 0153 0154 0155 0156 0157 0158	A121 A123 A126 A129 A12C A12F A133 A135 A137 A138 A13A	0C A6 B6 00 7E AA 7E A9 7E A0 7E B4 80 4F 4F C7 52 59 FF 04 5E 7E B2	00 1A B3 F6 4A			* * THESE  * BASIC LA13D	LDA JMP BYTES AF JMP JMP FCB FCB FCB FCB FDB JMP COMMAND FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRRSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) **RANDON SEED **RANDON SEED OF MANTISSA ** 811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR)
0144 0145 0146 0147 0148 0149 0150 0151 0152 0153 0154 0155 0156 0157 0158	A121 A123 A126 A129 A12C A12F A132 A133 A135 A137 A138 A13A	7E A9 7E A9 7E B4 80 4F 4F C7 52 59 FF 04 5E 7E B2 35 AA 66	00 1A B3 F6 4A			* * THESE  * BASIC LA13D LA13E	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS
0144 0145 0146 0147 0148 0149 > 0150 0151 0152 0153 0154 0155 0156 0157 0158 0159	A121 A123 A126 A129 A12C A12F A132 A133 A135 A137 A138 A13A A13D A140 A140 A142	7E A9 7E A9 7E A9 7E B4 80 4F 4F C7 52 59 FF 04 5E 7E B2 35 AA 66 AB 67 14	00 1A B3 F6 4A			* * THESE  * BASIC LA130 LA13E LA148 LA142	LDA JMP  BYTES AF JMP JMP JMP FCB FCB FCB FCB FCB FDB JMP COMMAND FCB FDB FDB FDB FDB FCB FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRGSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) **RANDON SEED **RANDON SEED OF MANTISSA **.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS
0144 0145 0146 0147 0148 0149 > 0150 0151 0152 0153 0154 0155 0156 0157 0158 0159 0161 0161	A121 A123 A126 A129 A12C A12F A133 A135 A137 A138 A13A A13A A13B A142 A143	7E A9 7E A9 7E A9 7E B4 80 4F 4F C7 52 59 FF 04 5E 7E B2 35 AA 66 AB 67 14 AB 1A	00 1A B3 F6 4A			* THESE  * BASIC LA13D LA13E LA14Ø LA142 LA142	LDA JMP  BYTES AF JMP JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20 LAB1A	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0149 > 0150 0151 0152 0153 0155 0156 0157 0158 0159 0160 0161 0162	A121 A123 A126 A129 A12C A12F A133 A135 A137 A138 A13A A13A A13B A142 A143	7E A9 7E A9 7E A9 7E B4 80 4F 4F C7 52 59 FF 04 5E 7E B2 35 AA 66 AB 67 14	00 1A B3 F6 4A			* THESE  * BASIC LA13D LA13E LA140 LA142 LA143 LA143 LA145	LDA JMP  BYTES AF JMP JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRGSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20 LAB1A LAB1A LAA29	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA **.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO JUMP TABLE FOR COMMANDS 20 BASIC SECONDARY COMMANDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0155 0155 0155 0156 0157 0168 0161 0162 0163	A121 A123 A126 A129 A122 A132 A133 A135 A137 A138 A13A A13D A13E A140 A142 A143 A145	7E A9 7E A9 7E B4 80 4F 752 59 FF 04 5E 7E B2 35 AA 66 AB 67 14 AB 1A AA 29	00 1A B3 F6 4A	4F 5		* * THESE  * BASIC LA13D LA13E LA148 LA142 LA143 LA145 * COPYRI	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRGSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20 LAB1A LAB1A LAA29	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0150 0151 0152 0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165	A121 A123 A126 A126 A127 A132 A133 A135 A137 A138 A13A A13D A13E A140 A142 A143 A145	0C A6 60 00 7E AA 7E A9 7E A9 4F C7 52 59 FF 04 5E 7E B2 35 AA 66 AB 67 14 AB 1A AA 29 43 4F 42 41	00 1A B3 F6 4A 77	49 4	2 20	* THESE  * BASIC LA130 LA13E LA142 LA143 LA143 LA144 * COPYRI LA147	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBBEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20 LAB1A LAA29 AGES	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0155 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167	A121 A123 A126 A127 A128 A127 A132 A133 A135 A137 A138 A138 A140 A142 A143 A144 A145 A147 A147 A147 A153	9C A6 86 90 7E AA 7E A9 7E B4 80 4F C7 52 59 82 35 AA 66 AB 1A AA 29 41 31 22 41 31 22	00 1A B3 F6 4A 77	49 4	2 20	* * THESE  * BASIC LA13D LA13E LA148 LA142 LA143 * COPYRI LA147	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROWHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM II 53 LAA66 LAB67 20 LAB1A LAA29 GAGES 'COLOR BASIC 1.2'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0155 0156 0157 0158 0169 0161 0162 0163 0164 0165 0166 0166 0166	A121 A123 A126 A129 A12C A132 A133 A135 A137 A138 A138 A134 A140 A142 A143 A145 A147 A140 A145 A153 A156	9C A6 86 97 7E A8 7E A9	00 1A B3 F6 4A 77	49 4	2 20 3 20	* THESE  * BASIC LA13D LA13E LA140 LA142 LA143 LA145 LA145 LA145 LA156	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRGSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20 LAB1A LAA29 AGGES 'COLOR BASIC 1.2'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0165 0165 0165	A121 A123 A126 A129 A12C A12F A133 A135 A137 A138 A138 A140 A142 A144 A143 A145 A147 A140 A156 A157	9C A6 86 90 7E AA 7E A9 7E A9 7E A9 7E B4 80 4F 67 7E B2 35 AA 66 AB 67 14 AA 29 41 31 26 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 42 41 31 26 41 31 26 41 31	00 1A B3 F6 4A 777	49 4 20 3	2 20 3 20 1 39	* THESE  * BASIC LA13D LA13E LA14Ø LA142 LA143 LA145 * COPYRI LA147	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROWHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM II 53 LAA66 LAB67 20 LAB1A LAA29 GAGES 'COLOR BASIC 1.2'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0156 0157 0158 0159 0161 0162 0163 0164 0165 0165 0165 0165	A121 A123 A126 A129 A12C A132 A133 A135 A137 A138 A134 A144 A142 A143 A145 A147 A145 A147 A153 A156 A157 A150	0C A6 86 00 7E AA 7E A9	00 1A B3 F6 4A 777	49 4 20 3	2 20 3 20 1 39	* THESE  * BASIC LA13D LA13E LA14Ø LA142 LA143 LA145 * COPYRI LA147	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRGSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20 LAB1A LAA29 AGGES 'COLOR BASIC 1.2'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0156 0157 0158 0160 0161 0162 0163 0164 0165 0166 0167 0166 0167	A121 A123 A126 A129 A12C A12F A132 A133 A135 A137 A138 A134 A144 A142 A143 A145 A147 A140 A157 A150 A157 A150 A157 A150 A157 A150 A157 A150 A157 A150 A157 A150 A157 A157 A157 A157 A157 A157 A157 A157	0C A6 86 00 7E AA 7E A9 7E A9 7E B4 80 4F C7 52 59 FF B2 35 AA 66 AB 67 14 AA 29 41 31 2E 41 31 2E 42 41 31 32 44 59 00 00 00 00 00 00 00 00 00 00 00 00 00	00 1A B3 F6 4A 777	49 4 20 3 54 4	2 20 3 20 1 39 1 4E	* THESE  * BASIC LA130 LA13E LA142 LA143 LA145 * COPYRI LA147  LA156 LA157	LDA JMP  BYTES AF JMP JMP FCB FCB FCB FCB FCB FDB FCB FCB FCB FCB FCC FCB FCC FCB FCC FCB FCC	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRGSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM I 53 LAA66 LAB67 20 LAB1A LAA29 AGGES 'COLOR BASIC 1.2'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0150 0151 0152 0153 0154 0155 0156 0157 0158 0160 0161 0162 0163 0164 0165 0166 0167 0168 0167 0168 0170 0171 0172	A121 A123 A126 A129 A12C A12F A132 A133 A135 A137 A138 A140 A142 A143 A145 A147 A140 A153 A156 A157 A150 A165 A166	0C A6 86 00 7E AA 7E A9 7E A9 80 4F FF 64 5E 82 5A A6 66 AB 67 14 AA 29 43 44 41 31 2E 60 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 44 59 64 64 64 64 64 64 64 64 64 64 64 64 64	00 1A B3 F6 4A 77 4C 53 32 29 20	49 4 20 3 54 4	2 20 3 20 1 39 1 4E	* THESE  * BASIC LA13D LA13E LA14B LA142 LA143 LA145 * COPYRI LA147  LA156 LA157	LDA JMP  BYTES AF JMP JMP FCB FCB FCB FCB FCB FDB FCB FCB FCB FCB FCC FCB FCC FCB FCC FCB FCC	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBBEL LB277 INTERPRETATION TABLE ROM II 53 LAA66 LAB67 20 LAB1A LAA29 AGES 'COLOR BASIC 1.2'  CR '(C) 1982 TANDY'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0155 0156 0157 0158 0161 0162 0163 0164 0165 0166 0167 0168 0167 0168 0179 0179 0171	A121 A123 A126 A129 A12C A132 A133 A135 A137 A138 A134 A144 A143 A144 A144 A143 A145 A153 A156 A157 A150 A156 A157 A158 A156 A167 A163 A163 A163 A163 A163 A163 A163 A163	9C A6 86 97 7E A8 7E A9	00 1A  B3 F6 4A  77  4C 53 32 29 20 43 54	49 4 20 3 54 4	2 2Ø 3 2Ø 1 39 1 4E	* THESE  * BASIC LA13D LA13E LA149 LA142 LA143 LA145 LA147 LA156 LA157	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBBEL LB277 INTERPRETATION TABLE ROM II 53 LAA66 LAB67 20 LAB1A LAA29 GAGES 'COLOR BASIC 1.2'  CR '(C) 1982 TANDY'  \$00 'MICROSOFT'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0159 0151 0152 0153 0154 0155 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0168 0167 0171 0172 0173	A121 A123 A126 A129 A12C A132 A133 A133 A133 A134 A134 A144 A143 A144 A143 A145 A153 A156 A157 A156 A157 A163 A163 A163 A163 A163 A163 A166	9C A6 86 97 7E A8 7E A9	00 1A  B3 F6 4A  77  4C 53 32 29 20 43 54	49 4 20 3 54 4	2 2Ø 3 2Ø 1 39 1 4E	* THESE  * BASIC LA130 LA13E LA142 LA143 LA145 * COPYRI LA147  LA156 LA157	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROMHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBBEL LB277 INTERPRETATION TABLE ROM II 53 LAA66 LAB67 20 LAB1A LAA29 GAGES 'COLOR BASIC 1.2'  CR '(C) 1982 TANDY'  \$00 'MICROSOFT'	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY COMMANDS POINTS TO SECONDARY FUNCTION RESERVED WORDS
0144 0145 0146 0147 0148 0150 0151 0152 0153 0154 0155 0156 0157 0158 0156 0161 0162 0163 0164 0165 0167 0168 0167 0170 0171 0172 0173 0174 0175 0177	A121 A123 A126 A129 A12C A132 A133 A135 A137 A138 A138 A140 A142 A143 A144 A143 A153 A156 A167 A163 A166 A167 A161 A161 A161 A161 A161 A161	ØC         A6           B6         ØØ           7E         A8           7E         A8           7E         A8           7E         B4           80         4F           52         59           FF         B4           8A         6A           AB         A1           AB         A1A	00 1A B3 F6 4A 777 4C 53 32 29 20 43 54	49 4 20 3 54 4	2 20 3 20 1 39 1 4E F 53	* THESE  * BASIC LA13D LA13E LA149 LA142 LA143 LA145 LA147 LA156 LA157	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROWHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM II 53 LAA66 LAB67 20 LAB1A LAA29 RAGES 'COLOR BASIC 1.2'  CR '(C) 1982 TANDY'  \$00 'MICROSOFT' CR,\$00 LA176	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *& 811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO JUMP TABLE FOR COMMANDS 2Ø BASIC SECONDARY FUNCTION RESERVED WORDS POINTS TO SECONDARY FUNCTION JUMP TABLE  GET A CHARACTER FROM CONSOLE IN
0144 0145 0146 0147 0148 0150 0151 0152 0153 0154 0155 0156 0157 0158 0159 0160 0161 0162 0163 0164 0165 0166 0167 0168 0169 0170 0170 0171 0172 0173 0174	A121 A123 A126 A129 A12C A132 A133 A135 A137 A138 A138 A140 A142 A143 A144 A143 A153 A156 A167 A163 A166 A167 A161 A161 A161 A161 A161 A161	0C A6 86 00 7E AA 7E A9 7E A9 7E A9 7E A9 7E B2 35 AA 66 AB 67 7E B2 35 AA 64 42 41 31 2E 60 84 45 44 45 44 45 44 45 45 46 46 46 60 00 00 00 00 00 00 00 00 00 00 00 00	00 1A B3 F6 4A 777 4C 53 32 29 20 43 54	49 4 20 3 54 4	2 20 3 20 1 39 1 4E F 53	* THESE  * BASIC LA13D LA13E LA14Ø LA143 LA145 * COPYRI LA147  LA156 LA157	LDA JMP  BYTES AF JMP JMP FCB	>0000 BROWHK  RE MOVED TO ADDRESSES \$10C- BIRQSV BFRQSV LB44A \$80 \$4FC7 \$5259 \$FF DEBDEL LB277 INTERPRETATION TABLE ROM II 53 LAA66 LAB67 20 LAB1A LAA29 RAGES 'COLOR BASIC 1.2'  CR '(C) 1982 TANDY'  \$00 'MICROSOFT' CR,\$00 LA176	IRQ SERVICE FIRQ SERVICE USR ADDRESS FOR 8K BASIC (INITIALIZED TO FC ERROR) *RANDOM SEED *RANDOM SEED OF MANTISSA *.811635157 UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER) KEYBOARD DEBOUNCE DELAY DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR) IMAGE 53 BASIC COMMANDS POINTS TO RESERVED WORDS POINTS TO RESERVED WORDS POINTS TO JUMP TABLE FOR COMMANDS 20 BASIC SECONDARY FUNCTION RESERVED WORDS POINTS TO SECONDARY FUNCTION JUMP TABLE

Ø179	A175	39			RTS		
0180							
0181				* CONSOLE			
Ø182	A176	BD Ø1		LA176	JSR	RVEC4	HOOK INTO RAM
Ø183 Ø184	A179	ØF 76		LA17B	CLR TST	CINBFL	RESET CONSOLE IN BUFFER FLAG = FULL
Ø185		27 32		LAI/B	BEQ	DEVNUM LA1B1	CHECK DEVICE NUMBER GØ DO CURSOR AND GET A KEY IF SCREEN MODE
Ø186		ØD 79			TST	CINCTR	TEST CHARACTER COUNTER
Ø187		26 Ø3			BNE	LA186	NOT EMPTY - READ IN SOME CASSETTE DATA
Ø188	A183	Ø3 70	ð		COM	CINBFL	SET TO \$FF: CONSOLE IN BUFFER EMPTY
Ø189	A185	39		LA185	RTS		
Ø19Ø Ø191	A186	34 74	,	* LA186	PSHS	II V V D	SAVE REGISTERS
Ø191		9E 7		LAIGO	LDX	U,Y,X,B CINPTR	PICK UP BUFFER POINTER
Ø193	A18A				LDA	,X+	GET NEXT CHAR
Ø194		34 Ø2				Á	SAVE CHAR ON STACK
Ø195	A18E				STX	CINPTR	SAVE NEW BUFFER POINTER
Ø196	A19Ø			LA19Ø	DEC	CINCTR	DECR CHAR COUNT
Ø197	A192				BNE	LA197	RETURN IF BUFFER NOT EMPTY
Ø198 Ø199	A194 A197	BD A6		LA197	JSR PULS	LA635 A,B,X,Y,U,PC	GO READ TAPE RESTORE REGISTERS
0200	NIJ,	33 10	,	*	1023	A, B, X, 1, 0, 10	RESTORE REGISTERS
0201	A199	ØA 94	1	LA199	DEC	BLKCNT	CURSOR BLINK DELAY
0202	A19B	26 Ø			BNE	LA1AB	NOT TIME FOR NEW COLOR
0203	A19D	C6 ØE			LDB	#11	*
0204	A19F				STB	BLKCNT	*RESET DELAY COUNTER
Ø2Ø5 Ø2Ø6	A1A1 A1A3				LDX LDA	CURPOS ,X	GET CURSOR POSITION GET CURRENT CURSOR CHAR
0207	A1A5				ADDA	#\$1Ø	BUMP TO NEXT COLOR
0208	A1A7	8A 8F			ORA	#\$8F	MAKE SURE IT S A SOLID GRAPHICS BLOCK
0209	A1A9				STA	, Х	STORE TO SCREEN
0210	A1AB			LA1AB	LDX	#DEBDEL	CURSOR BLINK DELAY
0211	A1AE	7E A7	7 D3	LA1AE	JMP	LA7D3	DELAY WHILE X DECREMENTS TO ZERO
Ø212 Ø213				* BLINK CI	IRSOR W	HILE WAITING FOR A KEYSTRO	KE
0214	A1B1	34 14	1	LA1B1		X,B	SAVE REGISTERS
0215	A1B3			LA1B3	BSR	LA199	GO DO CURSOR
0216	A1B5				BSR	KEYIN	GO CHECK KEYBOARD
0217		27 F			BEQ	LA1B3	LOOP IF NO KEY DOWN
Ø218 Ø219		C6 60	- ØØ 88		LDB STB	#\$60 [CURPOS]	BLANK BLANK CURRENT CURSOR CHAR ON SCREEN
0220	A1BF			LA1BF		B, X, PC	DEFINITE CONTROL CONTROL ON CONTEN
0221				*			
0222						ETS A KEYSTROKE FROM THE K	
Ø223 Ø224				* IS DOWN	. IT RE	TURNS ZERO TRUE IF THERE W	AS NO KEY DOWN.
Ø225	A1C1	7F FF	- Ø2	LA1C1	CLR	PIAØ+2	CLEAR COLUMN STROBE
0226	A1C4				LDA	PIAØ	READ KEY ROWS
Ø227	A1C7	43			COMA		COMPLEMENT ROW DATA
Ø228		48			ASLA		SHIFT OFF JOYSTICK DATA
Ø229	A1C9				BEQ	LA244	RETURN IF NO KEYS OR FIRE BUTTONS DOWN
Ø23Ø Ø231	A1CB	34 54		KEYIN	PSHS	U,X,B	SAVE REGISTERS
W231					LDII	#DTAØ	
Ø232	A1CD	CE F	ØØ		LDU LDX	#PIAØ #KEYBUF	POINT U TO PIAØ POINT X TO KEYBOARD MEMORY BUFFER
Ø232 Ø233	A1CD A1DØ		ØØ		LDU LDX CLRA	#PIAØ #KEYBUF	
	A1CD A1DØ	CE FF 8E Ø1 4F	ØØ		LDX		POINT X TO KEYBOARD MEMORY BUFFER
Ø233 Ø234 Ø235	A1CD A1DØ A1D3 A1D4 A1D5	CE FF 8E Ø1 4F 4A 34 12	= ØØ L 52		LDX CLRA DECA PSHS	#KEYBUF X,A	POINT X TO KEYBOARD MEMORY BUFFER * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA) * TO \$FF SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK
0233 0234 0235 0236	A1CD A1DØ A1D3 A1D4 A1D5 A1D7	CE FF 8E Ø1 4F 4A 34 12 A7 42	90 1 52 2		LDX CLRA DECA PSHS STA	#KEYBUF  X,A 2,U	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF
0233 0234 0235 0236 0237	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9	CE FF 8E Ø1 4F 4A 34 12 A7 42 69 42	90 1 52 2 2	LA1D9	LDX CLRA DECA PSHS STA ROL	#KEYBUF  X,A 2,U 2,U	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY
0233 0234 0235 0236 0237 0238	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB	CE FF 8E Ø1 4F 4A 34 12 69 42 24 43	- 00 1 52 2 2 2 3		LDX CLRA DECA PSHS STA ROL BCC	#KEYBUF  X,A 2,U 2,U LA220	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF
0233 0234 0235 0236 0237	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB A1DD	CE FF 8E Ø1 4F 4A 34 12 A7 42 69 42	- 00 L 52 2 2 2 2 3		LDX CLRA DECA PSHS STA ROL	#KEYBUF  X,A 2,U 2,U LA220	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE
0233 0234 0235 0236 0237 0238 0239 0240 0241	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB A1DD A1DF A1E1	CE FF 8E Ø1 4F 4A 34 12 A7 42 69 42 24 43 6C 6Ø 8D 59 A7 61	- 00 L 52 2 2 2 2 3 3		LDX CLRA DECA PSHS STA ROL BCC INC BSR STA	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA
0233 0234 0235 0236 0237 0238 0239 0240 0241	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB A1DD A1DF A1E1 A1E3	CE FF 8E Ø1 4F 4A 34 12 A7 42 69 42 24 43 6C 66 8D 59 A7 61 A8 84	- 00 L 52 2 2 2 3 3 9		LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S ,X	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA SET ANY BIT WHERE A KEY HAS MOVED
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB A1DD A1DF A1E1 A1E3 A1E5	CE FF 8E Ø1 4F 4A 34 12 69 42 24 43 6C 66 8D 59 A7 61 A8 84 A4 84	- 00 1 52 2 2 2 3 3 9 9		LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S ,X ,X	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB A1DD A1DF A1E1 A1E3 A1E5 A1E7	CE FF 8E Ø1 4F 4A 34 12 69 42 24 43 6C 66 8D 59 A7 61 A8 84 A4 84 E6 61	- 00 1 52 2 2 2 3 3 7 9 9		LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB	#KEYBUF  X,A 2,U 2,U LA220, ,S LA23A 1,S ,X ,X 1,S	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB A1DD A1DF A1E1 A1E3 A1E5	CE FF 8E Ø1 4F 4A 34 12 47 42 69 42 24 43 6C 6Ø 8D 59 A7 61 A8 84 84 84 86 61 E7 8Ø	- 00 1 52 2 2 2 3 3 7 9 9		LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S ,X ,X	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D8 A1DB A1DD A1DF A1E1 A1E3 A1E5 A1E7 A1E9 A1EB A1EC	CE FF 8E Ø1 4F 4A 34 12 43 6C 6Ø 8D 59 A7 61 A8 84 E6 61 E7 8Ø 4D 27 EE	= 00 1 52 2 2 2 3 3 9 1 1 1 1 1 1 1 1 3		LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB STB TSTA BEQ	#KEYBUF  X,A 2,U 2,U LA220, ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN
0233 0234 0235 0236 0237 0238 0239 0249 0241 0242 0243 0244 0245 0247 0248	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D8 A1DB A1DD A1DF A1E1 A1E3 A1E5 A1E7 A1E9 A1E8 A1EC A1EE	CE FF 8E Ø1 4F 4A 34 12 43 69 42 24 43 6C 66 8D 59 A7 61 A8 84 A4 84 E6 61 E7 86 42 E6 42 E6 42	= 00		LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB STB TSTA BEQ LDB	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X ,X ,LA23A 1,S ,X ,LA23A 1,S ,X ,X ,LA23A 1,S ,X	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <70 IF KEY WAS RELEASED GET NEW KEY DATA  STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D8 A1DB A1DD A1DF A1E1 A1E3 A1E5 A1E7 A1E9 A1E8 A1EC A1EE	CE FF 8E Ø1 4F 4A 34 12 43 6C 6Ø 8D 59 A7 61 A8 84 E6 61 E7 8Ø 4D 27 EE	= 00	LA1D9	LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB STB TSTA BEQ LDB STB	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0247 0247 0249 0250	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D8 A1DB A1DD A1DF A1E1 A1E3 A1E5 A1E7 A1E9 A1E8 A1EC A1EE	CE FF 8E Ø1 4F 4A 34 12 43 69 42 24 43 6C 66 8D 59 A7 61 A8 84 A4 84 E6 61 E7 86 42 E6 42 E6 42	= 00	LAID9  * THIS ROI	LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB STB TSTA BEQ LDB STB JTINE C	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S ,X ,X 1,S ,X 1,S ,X+ LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER  READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED  ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED  GET NEW KEY DATA  STORE IT IN KEY MEMORY  WAS A NEW KEY DOWN?  NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D8 A1DB A1DD A1DF A1E1 A1E3 A1E5 A1E7 A1E9 A1E8 A1EC A1EE	CE FF 8E Ø1 4F 4A 12 69 42 24 43 6C 66 8D 59 A7 61 A8 84 E6 61 E7 86 4D 27 EE E6 42 E7 62	= 00 1 52 2 2 2 2 3 3 7 9 9 1 1 1 1 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4	LAID9  * THIS ROI	LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB STB TSTA BEQ LDB STB JTINE C	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER  READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED  ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED  GET NEW KEY DATA  STORE IT IN KEY MEMORY  WAS A NEW KEY DOWN?  NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247 0248 0249 0249	A1CD A1DØ A1D3 A1D4 A1D5 A1D7 A1D9 A1DB A1DF A1E1 A1E3 A1E5 A1E5 A1E6 A1E7 A1E8 A1EC A1EE A1FØ	CE FF 8E Ø1 4F 4A 4A 4A 4A 4A 8A 8A 5A 6A	= 00	LAID9  * THIS ROI	LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB STB TSTA BEQ LDB STB JTINE 60 IN A	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S ONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN
0233 0234 0235 0236 0237 0239 0240 0241 0242 0243 0244 0245 0247 0246 0247 0248 0249 0251 0253	A1CD A1DØ A1DØ A1DØ A1DF A1DF A1DF A1DB A1DB A1DB A1DB A1EF A1EF A1EF A1EF A1EF A1EF A1EF A1EF	CE FF 82 81 81 81 81 81 81 81 81 81 81 81 81 81	7 00 1 5 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	LA1D9  * THIS ROU * FROM 0-	LDX CLRA DECA PSHS STA ROL BCC INC BSR STA EORA ANDA LDB STB TSTA BEQ LDB STB JTINE (50 IN A LDB LDB LSRA	#KEYBUF  X,A 2,U 2,U LA220, ,S LA23A 1,S, ,X ,X LA23D 1,S ,X ,X ,X ,X ,X LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$Ø8	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE TO \$FF  * ROTATE COLUMN COUNTER INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED  ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED  GET NEW KEY DATA  STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8  ADD 8 FOR EACH ROW OF KEYBOARD  ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW
0233 0234 0235 0236 0237 0238 0239 0240 0242 0243 0244 0245 0245 0246 0247 0248 0249 0251 0252	A1CD A1D0 A1D3 A1D4 A1D5 A1D7 A1D9 A1D8 A1DD A1D7 A1E1 A1E3 A1E5 A1E7 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8	CE FI E E E E E E E E E E E E E E E E E E	= 00	LA1D9  * THIS ROU * FROM 0-	LDX CLRA DECA PSHS STA ROL BCC INC STA EORA ANDA LSTB TSTA BEQ LDB STB TSTA BEQ LDB STB TITLE LDB ADDB ADDB ADDB ADDB ADDB ADDB ADDB	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S ONVERTS THE KEY DEPRESSION (CCG CORRESPONDING TO THE K #\$F8 #\$98 LA1F4	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA=0 IF NO NEW KEY DOWN, <70 IF KEY WAS RELEASED GET NEW KEY DATA  STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB=0 AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247 0248 0249 0250 0253 0253	A1CD A1DØ A1DØ A1DØ A1DF A1DF A1DF A1DB A1DB A1DB A1DB A1EF A1EF A1EF A1EF A1EF A1EF A1EF A1EF	CE FI E E E E E E E E E E E E E E E E E E	= 00	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA DECA DECA PSHS STA ROL BCC BSR STA EORA ANDA LDB STB TSTB TSTB TSTB TSTB TSTB TSTB TST	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$Ø8  LA1F4 ,S	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA-Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB-Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER
0233 0234 0235 0236 0237 0238 0249 0241 0241 0242 0243 0244 0245 0246 0247 0248 0245 0250 0251	A1CD A1DØ A1DØ A1D3 A1D4 A1D5 A1D7 A1D8 A1D7 A1D8 A1D7 A1E1 A1E3 A1E7 A1E9 A1E6 A1E7 A1E8 A1E6 A1E7 A1E8 A1E7 A1E8 A1E7 A1E8 A1E7 A1E8 A1E7 A1E8 A1E7 A1E8 A1E7 A1E8 A1E8 A1E7 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8	CE FI	= 00	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA DECA PSHS STA ROL BSC INC BSR STB ECRA ANDA LDB TSTA BEQ LDB LSRA BLUB LSRA BBCC LDB LSRA ANDA LDB LSRA BCC LDB LSRA ANDA LDB LSRA ANDB LSRA	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$F8 LA1F4 ,S LE VALUE IN ACCB INTO ASCII	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED  ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED  GET NEW KEY DATA  STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER  EY THAT WAS DOWN  TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8  ADD 8 FOR EACH ROW OF KEYBOARD  ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG  ADD IN THE COLUMN NUMBER
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247 0248 0249 0250 0253 0253	A1CD A1D0 A1D3 A1D4 A1D5 A1D7 A1D9 A1D8 A1DD A1D7 A1E1 A1E3 A1E5 A1E7 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8 A1E8	CE FIFE SERVICE SERVIC	= 00	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA DECA DECA PSHS STA ROL BCC BSR STA EORA ANDA LDB STB TSTB TSTB TSTB TSTB TSTB TSTB TST	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$Ø8  LA1F4 ,S	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA-Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB-Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER
0233 0234 0235 0236 0237 0238 0249 0240 0241 0242 0243 0244 0245 0246 0247 0248 0250 0251 0252 0253 0254 0257 0258 0257	A1CD A1DW A1D3 A1D4 A1D5 A1D7 A1D8 A1D8 A1D9 A1D8 A1D7 A1E1 A1E3 A1E7 A1E9 A1E8 A1EC A1FW A1FW A1FW A1FW A1FW A1FW A1FW A1FW	CE FIFE  RE 01  4A 1  4A 1  4A 1  4A 2  4A 4  4A 6C 66  6A 6C 6A 6C  6A 6C 6A 6C  CB 6C 6A 44  CC 6A 6C  CB 6C 6A 44  CC 6A 6C  CC 6C 6C  CC 6C 6C  CC 7 41  CC 7 41  CC 1 11	= 00	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA PSHS STA ROL BSR CI INC BSR EORA ANDA LDB TSTA BEQ LDB LSRA BEQ LDB ADDB LSRA ADDB LSRA ADDB LSRA BEQ COMPB BHI	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCB CORRESPONDING TO THE K #\$F8 #\$98  LA1F4 ,S LE VALUE IN ACCB INTO ASCII LA245 #26 LA247	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER  THE AT SIGN KEY WAS DOWN WAS IT A LETTER? NO
0233 0234 0235 0236 0237 0238 0249 0241 0242 0243 0244 0245 0247 0248 0247 0248 0255 0255 0255 0255 0255	A1CD A1DM A1D3 A1D4 A1D5 A1D7 A1D8 A1DD A1DB A1DB A1E1 A1E3 A1E5 A1E7 A1E8 A1E6 A1F6 A1F6 A1F7 A1F8 A1F7 A1F8 A1F7 A1F8 A1F7 A1F8 A1F7 A1F8 A1F7 A1F8 A1F7 A1F8 A1F7 A1F8 A1F7 A1F8 A1F8 A1F8 A1F8 A1F8 A1F8 A1F8 A1F8	CE FIFE SE 81 81 81 81 81 81 81 81 81 81 81 81 81	= 00	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA DECA PSHS STA BCC INC BSR EORA ANDA LDB STB EORA ANDA LDB STB LDB STB LDB ADDB ADDB BCQ CMPB BEQ CMPB BHI ORB	#KEYBUF  X,A 2,U 2,U LA220,S LA23A 1,S ,X ,X ,X LA109 2,U 2,S CONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$88  LA1F4 ,S LE VALUE IN ACCB INTO ASCII LA245 #26 LA247 #\$40	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE TO \$FF  * ROTATE COLUMN COUNTER INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA=0 IF NO NEW KEY DOWN, <70 IF KEY WAS RELEASED GET NEW KEY DATA  STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN TO MAKE SURE ACCB=0 AFTER FIRST ADDB #8 ADD B FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER  THE AT SIGN KEY WAS DOWN WAS IT A LETTER? NO YES, CONVERT TO UPPER CASE ASCII
0233 0234 0235 0236 0237 0238 0239 0240 0242 0243 0244 0245 0245 0246 0247 0248 0250 0252 0253 0254 0255 0255 0255	A1CD A1DM A1D3 A1D3 A1D4 A1D5 A1D7 A1D8 A1D8 A1DB A1DB A1DB A1DB A1E1 A1E3 A1E5 A1E7 A1E8 A1E7 A1E8 A1F0 A1F8 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F7 A1F1 A1F1 A1F1 A1F1 A1F1 A1F1 A1F1	CE FIFE SE	= 00	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA PSHS STA RDCC INC BSR EORA ANDA STB EORA ANDA STB LDB STB STB LDB STB STB STB STB STB STB STB STB STB ST	#KEYBUF  X,A 2,U 2,U LA220,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S ONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$08  LA1F4 ,S LE VALUE IN ACCB INTO ASCII LA245 #26 LA247 #\$48 LA22E	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DOWN? MAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER  THE AT SIGN KEY WAS DOWN WAS IT A LETTER? NO YES, CONVERT TO UPPER CASE ASCII CHECK FOR THE SHIFT KEY
0233 0234 0235 0236 0237 0238 0249 0240 0241 0242 0243 0244 0245 0246 0247 0246 0247 0250 0251 0252 0253 0254 0255 0257 0256 0257 0258 0259 0260 0261	A1CD A1DB A1DB A1DB A1DB A1DB A1DB A1DB A1D	CE FIFE  RE 01  4A 1  4A 1  4A 1  4A 2  4A 4  69 42  44 4  60 65  60 65  A7 61  A8 84  84 84  66 67 86  67 86  44 4  CE 66 42  CE 7 46  CE 86  CE 86  CE 86  CE 86  CE 42  CE 44  CE 61  CE 86	7 00 1 52 2 2 2 3 3 3 4 4 4 4 4 4 5 5 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA PSHS STA ROL BSR CLINC BSR EORA ANDA LDB STB TSTA BEQ LDB STB STB STB STB STB STB STB STB STB ST	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$68  LA1F4 ,S LE VALUE IN ACCB INTO ASCII LA245 #26 LA247 #\$40 LA22E CASFLG	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA-Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DATA STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB-Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER  THE AT SIGN KEY WAS DOWN WAS IT A LETTER?  NO YES, CONVERT TO UPPER CASE ASCII CHECK FOR THE SHIFT KEY  * OR IN THE CASE FLAG & BRANCH IF IN UPPER
0233 0234 0235 0236 0237 0238 0239 0240 0242 0243 0244 0245 0245 0246 0247 0248 0250 0252 0253 0254 0255 0255 0255	A1CO A1DW A1D3 A1D4 A1D5 A1D7 A1D8 A1D8 A1DB A1DB A1DB A1E1 A1E3 A1E7 A1E9 A1E8 A1E7 A1F0 A1F6 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F7 A1F9 A1F8 A1F8 A1F8 A1F8 A1F8 A1F8 A1F8 A1F8	CE FIFE SE	7 00 1 5 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA PSHS STA RDCC INC BSR EORA ANDA STB EORA ANDA STB LDB STB STB LDB STB STB STB STB STB STB STB STB STB ST	#KEYBUF  X,A 2,U 2,U LA220,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S ONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$08  LA1F4 ,S LE VALUE IN ACCB INTO ASCII LA245 #26 LA247 #\$48 LA22E	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED GET NEW KEY DOWN? MAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER  THE AT SIGN KEY WAS DOWN WAS IT A LETTER? NO YES, CONVERT TO UPPER CASE ASCII CHECK FOR THE SHIFT KEY
0233 0234 0235 0236 0237 0238 0239 0240 0241 0242 0243 0244 0245 0246 0247 0246 0247 0248 0259 0250 0251 0252 0253 0256 0257 0258 0259 0260 0261 0262	A1CD A1DB A1DB A1DB A1DB A1DB A1DB A1DB A1D	CE FIFE  CE SE OF	7 00 1 52 2 2 2 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA PSHS STA ROL BSR STA EORA ANDA LDB STSTB LDB STB STB LDB STB STB STB COMP LDB ADDB LSRA ADDB STB STB STB STB	#KEYBUF  X,A 2,U 2,U LA220 ,S LA23A 1,S ,X ,X 1,S ,X+ LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCB CORRESPONDING TO THE K #\$F8 #\$08  LA1F4 ,S LE VALUE IN ACCB INTO ASCII LA245 #26 LA247 #\$40 LA247 #\$40 LA22E CASFLG LA22C #\$20 ,S	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * NOTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA TEMP STORE KEY DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA-Ø IF NO NEW KEY DOWN, <70 IF KEY WAS RELEASED GET NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB-Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER  THE AT SIGN KEY WAS DOWN WAS IT A LETTER? NO CHECK FOR THE SHIFT KEY  * OR IN THE CASE FLAG & BRANCH IF IN UPPER  * CASE MODE OR SHIFT KEY  * OR IN THE CASE FLAG & BRANCH IF IN UPPER  * CASE MODE OR SHIFT KEY DOWN CONVERT TO LOWER CASE TEMP STORE ASCII VALUE
0233 0234 0235 0237 0238 0239 0240 0242 0243 0244 0245 0246 0247 0250 0250 0255 0256 0257 0258 0259 0260 0261 0262 0263	A1CD A1DB A1DB A1DB A1DB A1DB A1DB A1DB A1D	CE FIFE SE	7 00 1 52 2 2 2 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4	* THIS ROU * FROM Ø-! LA1F4	LDX CLRA PSHS STA RBCC INC BSR STA EORA ANDA STB STB STB STB STB LDB LDB STB LDB STB LDB STB LDB STB LDB STB LDB STB LDB STB LDB STB STB STB STB STB STB STB STB STB ST	#KEYBUF  X,A 2,U 2,U LA22Ø ,S LA23A 1,S ,X ,X ,X ,X LA1D9 2,U 2,S CONVERTS THE KEY DEPRESSION CCCB CORRESPONDING TO THE K #\$F8 #\$68  LA1F4 ,S LA245 #26 LA247 #\$448 LA22E CASFLG LA247 #\$48	POINT X TO KEYBOARD MEMORY BUFFER  * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)  * TO \$FF  SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK INITIALIZE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE TO \$FF  * ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY  * INTO BIT Ø - BRANCH IF 8 SHIFTS DONE INCREMENT COLUMN COUNTER READ KEYBOARD ROW DATA  SET ANY BIT WHERE A KEY HAS MOVED ACCA=0 IF NO NEW KEY DOWN, <70 IF KEY WAS RELEASED GET NEW KEY DATA  STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? NO-CHECK ANOTHER COLUMN  * GET COLUMN STROBE DATA AND  * TEMP STORE IT ON THE STACK INTO A NUMBER EY THAT WAS DOWN  TO MAKE SURE ACCB=0 AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA HAS THE ROW NUMBER OF THIS KEY - ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG ADD IN THE COLUMN NUMBER  THE AT SIGN KEY WAS DOWN WAS IT A LETTER?  NO YES, CONVERT TO UPPER CASE ASCII CHECK FOR THE SHIFT KEY  * OR IN THE CASE FLAG & BRANCH IF IN UPPER  * CASE MODE OR SHIFT KEY DOWN CONVERT TO LOWER CASE

Ø268	A211	8D 9B			BSR	LA1AE	
Ø269		C6 FF			LDB	#\$FF	SET COLUMN STROBE TO ALL ONES (NO
0270	A215	8D 21			BSR	LA238	STROBE) AND READ KEYBOARD
0271	A217	4 C			INCA		= INCR ROW DATA, ACCA NOW Ø IF NO JOYSTICK
0272		26 Ø6			BNE	LA220	= BUTTON DOWN. BRANCH IF JOYSTICK BUTTON DOWN
Ø273 Ø274		E6 62		LA21A	LDB	2,5	GET COLUMN STROBE DATA READ A KEY
0274 0275		8D 1A A1 61			BSR CMPA	LA238 1,S	IS IT THE SAME KEY AS BEFORE DEBOUNCE?
Ø276	A22Ø	35 12		LA22Ø	PULS	A, X	REMOVE TEMP SLOTS FROM THE STACK AND RECOVER
Ø277				*		ŕ	THE ASCII VALUE OF THE KEY
Ø278	A222	26 Ø7			BNE	LA22B	NOT THE SAME KEY OR JOYSTICK BUTTON
Ø279		81 12			CMPA	#\$12	IS SHIFT ZERO DOWN?
0280	A226	26 Ø4			BNE	LA22C	NO
Ø281	A228	73 Ø1	1A	I A 22D	COM	CASFLG	YES, TOGGLE UPPER CASE/LOWER CASE FLAG
Ø282 Ø283	A22B	4F 35 D4		LA22B LA22C	CLRA PULS	B,X,U,PC	SET ZERO FLAG TO INDICATE NO NEW KEY DOWN RESTORE REGISTERS
Ø284	ALLO	33 D4		ENEEO	1023	5, 7, 0, 10	RESTORE REGISTERS
Ø285				* TEST FO	R THE S	HIFT KEY	
Ø286	A22E	86 7F		LA22E	LDA	#\$7F	COLUMN STROBE
Ø287	A23Ø	A7 42			STA	2,U	STORE TO PIA
Ø288		A6 C4			LDA	,U	READ KEY DATA
Ø289 Ø29Ø	A234	43 84 4Ø			COMA	44.40	*
Ø290 Ø291	A235 A237	39			ANDA RTS	#\$40	* SET BIT 6 IF SHIFT KEY DOWN RETURN
0292	ALS7	33			KIS		KETOKH
Ø293				* READ TH	E KEYBO	ARD	
0294	A238	E7 42		LA238	STB	2,U	SAVE NEW COLUMN STROBE VALUE
Ø295	A23A	A6 C4		LA23A	LDA	, U	READ PIAØ, PORT A TO SEE IF KEY IS DOWN
Ø296				*			A BIT WILL BE ZERO IF ONE IS
Ø297	A23C	8A 8Ø 6D 42			ORA	#\$80	MASK OFF THE JOYSTICK COMPARATOR INPUT
Ø298 Ø299	A23E A24Ø	2B Ø2			TST BMI	\$02,U LA244	ARE WE STROBING COLUMN 7?
0300		8A CØ			ORA	#\$CØ	YES, FORCE ROW 6 TO BE HIGH - THIS WILL CAUSE
0301	712.12	07. 02		*	0101	7,400	THE SHIFT KEY TO BE IGNORED
0302	A244	39		LA244	RTS		RETURN
0303							
0304		C6 33		LA245	LDB	#51	CODE FOR AT SIGN
0305		8E A2	38	LA247	LDX	#CONTAB-\$36	POINT X TO CONTROL CODE TABLE
Ø3Ø6 Ø3Ø7		C1 21 25 16			CMPB BLO	#33 LA264	KEY NUMBER <33? YES (ARROW KEYS, SPACE BAR, ZERO)
Ø3Ø8		8E A2			LDX	#CONTAB-\$54	POINT X TO MIDDLE OF CONTROL TABLE
0309		C1 3Ø			CMPB	#48	KEY NUMBER >48?
0310		24 ØF			BHS	LA264	YES (ENTER, CLEAR, BREAK, AT SIGN)
Ø311	A255	8D D7			BSR	LA22E	CHECK SHIFT KEY (ACCA WILL CONTAIN STATUS)
Ø312	A257	C1 2B			CMPB	#43	IS KEY A NUMBER, COLON OR SEMICOLON?
0313	A259	23 Ø2			BLS	LA25D	YES
Ø314 Ø315	A25B	88 40			EORA	#\$40	TOGGLE BIT 6 OF ACCA WHICH CONTAINS THE SHIFT DATA
Ø316	A25D	4 D		LA25D	TSTA		ONLY FOR SLASH, HYPHEN, PERIOD, COMMA SHIFT KEY DOWN?
Ø317	A25E	26 AC		2,1200	BNE	LA2ØC	YES
Ø318	A26Ø	CB 10			ADDB	#\$10	NO, ADD IN ASCII OFFSET CORRECTION
Ø319	A262	2Ø A8			BRA	LA2ØC	GO CHECK FOR DEBOUNCE
0320	A264	58		LA264	ASLB		MULT ACCB BY 2 - THERE ARE 2 ENTRIES IN CONTROL
Ø321 Ø322	A265	8D C7		*	BSR	LA22E	TABLE FOR EACH KEY - ONE SHIFTED, ONE NOT
Ø322		27 Ø1			BEQ		CHECK SHIFT KEY
Ø324	A269						NOT DOWN
Ø325		5C				LA26A	NOT DOWN ADD ONE TO GET THE SHIFTED VALUE
Ø326	A26A	5C E6 85		LA26A	INCB LDB	B,X	NOT DOWN ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE
					INCB		ADD ONE TO GET THE SHIFTED VALUE
Ø327		E6 85		LA26A	INCB LDB	В,Х	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE
Ø327 Ø328		E6 85		*	INCB LDB BRA	B,X LA2ØC	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE
Ø327 Ø328 Ø329	A26C	E6 85 20 9E		* * * CONTROL	INCB LDB BRA	B,X LA20C UNSHIFTED, SHIFTED VALUE	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE ES
Ø327 Ø328 Ø329	A26C A26E	E6 85 20 9E		*	INCB LDB BRA	B,X LA2ØC	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE
0327 0328 0329 0330 0331 0332	A26C A26E A27Ø A272	E6 85 20 9E 5E 5F ØA 5B Ø8 15		* * * CONTROL	INCB LDB BRA TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES UP ARROW
0327 0328 0329 0330 0331 0332 0333	A26E A27Ø A272 A274	5E 5F ØA 5B Ø8 15		* * * CONTROL	TABLE FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW
0327 0328 0329 0330 0331 0332 0333	A26E A27Ø A272 A274 A276	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø		* * * CONTROL	TABLE FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR
0327 0328 0329 0330 0331 0332 0333 0334 0335	A26E A27Ø A272 A274 A276 A278	5E 5F ØA 5B Ø8 15 Ø9 5D 20 20 30 12		* * * CONTROL	INCB LDB BRA TABLE FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO
0327 0328 0329 0330 0331 0332 0333 0334 0335	A26C A26E A27Ø A272 A274 A276 A278 A27A	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD		* * * CONTROL	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$0D	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER
0327 0328 0329 0330 0331 0332 0333 0334 0335 0336	A26C A26E A27Ø A272 A274 A276 A278 A27A A27C	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD ØC 5C		* * * CONTROL	TABLE FCB FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$09,\$5D \$20,\$20 \$30,\$12 \$0D,\$0D \$0C,\$5C	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR
0327 0328 0329 0330 0331 0332 0333 0334 0335	A26C A26E A27Ø A272 A274 A276 A278 A27A A27C A27E	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD		* * * CONTROL	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$FF \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$6C \$03,\$03	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK
0327 0328 0329 0330 0331 0332 0333 0334 0335 0336 0337	A26C A26E A27Ø A272 A274 A276 A278 A27A A27C A27E	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD ØC 5C Ø3 Ø3		* * * CONTROL	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$09,\$5D \$20,\$20 \$30,\$12 \$0D,\$0D \$0C,\$5C	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR
0327 0328 0329 0330 0331 0332 0333 0334 0335 0336 0337 0338	A26E A27Ø A272 A274 A276 A278 A27A A27C A27E A28Ø	5E 5F ØA 5B ØB 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD ØC 5C Ø3 Ø3 4Ø 13		* * CONTROL CONTAB  * CONSOLE	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$0D \$0C,\$5C \$03,\$03 \$40,\$13	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN
0327 0328 0329 0331 0332 0333 0334 0335 0336 0337 0338 0339 0341 0342	A26C  A26E A27Ø A272 A274 A276 A278 A27A A27C A27E A28Ø	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø ØD ØD ØC 5C Ø3 Ø3 4Ø 13	67	* * CONTROL CONTAB	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$5C \$03,\$03 \$40,\$13	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN
0327 0328 0329 0331 0332 0333 0334 0335 0336 0337 0338 0339 0341 0342 0343	A26C  A26E A270 A272 A274 A276 A278 A27A A27C A27E A280  A282 A285	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD ØC 5C Ø3 Ø3 4Ø 13	67	* * CONTROL CONTAB  * CONSOLE	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$0D \$0C,\$5C \$03,\$03 \$40,\$13	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB
0327 0328 0329 0330 0331 0332 0333 0334 0335 0336 0337 0338 0339 0340 0341 0342	A26C  A26E A27Ø A272 A274 A276 A278 A277 A27C A27E A28Ø  A282 A285 A287	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD ØC 5C Ø3 Ø3 4Ø 13 BD Ø1 34 Ø4 D6 6F	67	* * CONTROL CONTAB  * CONSOLE	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$5C \$03,\$03 \$40,\$13	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER
0327 0328 0329 0330 0331 0332 0333 0334 0335 0336 0337 0340 0341 0342 0343 0344 0345	A26C  A26E A27Ø A272 A274 A276 A278 A27C A27E A28Ø  A282 A285 A287 A289	5E 5F ØA 5B Ø8 15 Ø9 5D Ø2 20 20 Ø ØD Ø	67	* * CONTROL CONTAB  * CONSOLE	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$FF \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$0D \$0C,\$6C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS
0327 0328 0329 0330 0331 0332 0333 0334 0335 0336 0337 0338 0339 0340 0341 0342	A26C  A26E A270 A272 A274 A276 A278 A27C A27E A280  A282 A285 A287 A289 A28A	5E 5F ØA 5B Ø8 15 Ø9 5D 2Ø 2Ø 3Ø 12 ØD ØD ØC 5C Ø3 Ø3 4Ø 13 BD Ø1 34 Ø4 D6 6F	67	* * CONTROL CONTAB  * CONSOLE	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$FF \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$0D \$0C,\$6C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER
0327 0328 0329 0330 0331 0332 0333 0334 0335 0336 0337 0341 0342 0343 0341 0342	A26C  A26E A270 A272 A274 A276 A278 A27A A27C A27E A280  A282 A285 A287 A288	5E 5F ØA 5B Ø8 15 Ø9 5D 20 20 20 20 20 00 00 5C Ø3 Ø3 40 13 BD Ø1 34 Ø4 D6 6F 5C 35 Ø4	67	* * CONTROL CONTAB  * CONSOLE	TABLE FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$5C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM B	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB
0327 0328 0329 0330 0331 0332 0333 0334 0336 0337 0338 0340 0341 0342 0343 0344 0344 0344 0344	A26C  A26E A270 A272 A274 A276 A278 A277 A27E A280  A282 A285 A287 A289 A282 A288	E6 85 20 9E 5E 5F 6A 5B 68 15 69 5D 60 6 6F 5C 64 5E 6	67	* * CONTROL CONTAB  * CONSOLE	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	B,X LA20C  UNSHIFTED, SHIFTED VALUE \$5E,\$FF \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$5C \$03,\$03 \$40,\$13  RVEC3 B DEVNUM B LA2BF LA30A TE	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB SEND TO LINE PRINTER SEND TO SCREEN
0327 0328 0329 0330 0331 0332 0333 0334 0335 0337 0338 0340 0341 0342 0343 0344 0345 0347 0348	A26C  A26E A270 A272 A274 A276 A277 A277 A276 A280  A282 A285 A287 A289 A282 A286 A286 A286	E6 85 20 9E 5E 5F 6A 5B 68 15 69 5D 60 6C 5C 60 40 134 64 134 64 134 64 134 64 134 64 134 65 67 7A 34 16 7A 14 14 16 7A	67	* * CONTROL CONTAB  * CONSOLE PUTCHR	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$0D,\$0D \$0C,\$5C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM B LA2BF LA30A TE X,B,A	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB SEND TO LINE PRINTER SEND TO SCREEN RESTORE REGISTERS
0327 0328 0329 0330 0331 0331 0333 0334 0335 0336 0337 0348 0341 0342 0343 0344 0345 0346 0347 0348 0347 0348	A26C  A26E A270 A272 A274 A276 A278 A278 A270 A278 A280  A282 A285 A287 A289 A282 A286 A286 A286 A292	E6 85 20 9E 5E 5F 6A 5B 6B 15 69 5D 6D	67	* * CONTROL CONTAB  * CONSOLE PUTCHR	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	B,X LA20C  UNSHIFTED, SHIFTED VALUE \$5E,\$FF \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$5C \$03,\$03 \$40,\$13  RVEC3 B DEVNUM B LA2BF LA30A TE	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB SEND TO LINE PRINTER SEND TO SCREEN  RESTORE REGISTERS GET FILE STATUS
0327 0328 0329 0330 0331 0331 0332 0333 0334 0335 0336 0337 0338 0341 0342 0343 0344 0345 0346 0347 0348	A26C  A26E A278 A272 A274 A276 A278 A27A A27C A288 A288 A288 A288 A288 A288 A288 A28	E6 85 20 9E 5E 5F 6A 5B 88 15 69 5D 6C 5C 5F 6C 35 64 34 16 75 66 78 5A	67	* * CONTROL CONTAB  * CONSOLE PUTCHR	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$FF \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$0D \$0C,\$5C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM B LA2BF LA30A TE X,B,A FILSTA	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB SEND TO LINE PRINTER SEND TO SCREEN  RESTORE REGISTERS GET FILE STATUS INPUT FILE?
0327 0328 0329 0330 0331 0332 0333 0334 0336 0337 0338 0340 0341 0342 0343 0344 0345 0346 0347 0346 0347 0348 0349 0350 0351	A26C  A26E  A270 A272 A274 A276 A278 A278 A278 A288  A282 A285 A287 A288 A288 A288 A288 A298 A294 A295	E6 85 20 9E 6 9E	67	* * CONTROL CONTAB  * CONSOLE PUTCHR	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$5C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM B LA2BF LA30A TE X,B,A FILSTA LA2A6	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB SEND TO LINE PRINTER SEND TO SCREEN  RESTORE REGISTERS GET FILE STATUS INPUT FILE? YES
0327 0328 0329 0330 0331 0331 0332 0333 0334 0335 0336 0337 0338 0341 0342 0343 0344 0345 0346 0347 0348	A26C  A26E  A270 A272 A274 A276 A278 A278 A278 A288  A282 A285 A287 A288 A288 A288 A288 A298 A294 A295	E6 85 20 9E 5E 5F 6A 5B 68 15 69 5D 60 5G 33 63 40 13 66 5C 7A 5B 6A 5B	67	* * CONTROL CONTAB  * CONSOLE PUTCHR	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$FF \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$0D \$0C,\$5C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM B LA2BF LA30A TE X,B,A FILSTA	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB SEND TO LINE PRINTER SEND TO SCREEN  RESTORE REGISTERS GET FILE STATUS INPUT FILE?
0327 0328 0329 0330 0331 0331 0333 0334 0335 0336 0337 0348 0341 0342 0343 0344 0345 0347 0348 0347 0348 0349 0353	A26C  A26E A278 A272 A274 A276 A278 A27A A276 A289 A280 A282 A285 A289 A280 A280 A292 A294 A295 A299 A299	E6 85 20 9E 5E 5F 6A 5B 68 15 69 5D 60 5G 33 63 40 13 66 5C 7A 5B 6A 5B	67	* * CONTROL CONTAB  * CONSOLE PUTCHR	INCB LDB BRA TABLE FCB FCB FCB FCB FCB FCB FCB FCB FCB FCB	B,X LA20C UNSHIFTED, SHIFTED VALUE \$5E,\$5F \$0A,\$5B \$08,\$15 \$09,\$5D \$20,\$20 \$30,\$12 \$00,\$00 \$0C,\$5C \$03,\$03 \$40,\$13 RVEC3 B DEVNUM B LA2BF LA30A TE X,B,A FILSTA LA2A6	ADD ONE TO GET THE SHIFTED VALUE GET ASCII CODE FROM CONTROL TABLE GO CHECK DEBOUNCE  ES  UP ARROW DOWN ARROW RIGHT ARROW LEFT ARROW SPACE BAR ZERO ENTER CLEAR BREAK AT SIGN  HOOK INTO RAM SAVE ACCB GET DEVICE NUMBER SET FLAGS RESTORE ACCB SEND TO LINE PRINTER SEND TO SCREEN  RESTORE REGISTERS GET FILE STATUS IMPUT FILE? YES TEMP CHAR CTR

Ø357	A29C	8D ØA			BSR	LA2A8	YES, WRITE DATA BLOCK TO TAPE
Ø358	A29E	9E 7A		LA29E	LDX	CINPTR	GET BUFFER POINTER
Ø359	A2AØ	A7 8Ø			STA	, X+	PUT CHAR IN CASSETTE BUFFER
Ø36Ø Ø361	A2A2 A2A4	9F 7A ØC 79			STX INC	CINPTR CINCTR	STORE NEW BUFFER POINTER INCR BYTE COUNT
Ø362	A2A6	35 96		LA2A6	PULS	A,B,X,PC	THOR BITE GOOK!
Ø363							
Ø364 Ø365	A2A8	C6 Ø1		* WRITE A LA2A8	BLOCK LDB	OF DATA TO TAPE #1	DATA BLOCK TYPE - NOT A HEADER BLOCK
Ø366	A2AA	D7 7C		LAZAG LAZAA	STB	BLKTYP	BLOCK NUMBER
Ø367	A2AC	8E Ø1	DA		LDX	#CASBUF	CASSETTE BUFFER
Ø368	A2AF	9F 7E			STX	CBUFAD	STARTING ADDRESS
Ø369 Ø37Ø	A2B1 A2B3	D6 79 D7 7D			LDB STB	CINCTR BLKLEN	GET NUMBER OF BYTES BYTE COUNT
Ø371	A2B5	34 62			PSHS	U,Y,A	SAVE REGISTERS
Ø372	A2B7	BD A7	E5		JSR	LA7E5	WRITE A BLOCK ON TAPE
Ø373	A2BA	35 62				A,Y,U	RESTORE REGISTERS
Ø374 Ø375	A2BC	7E A6	50		JMP	LA65Ø	RESET BUFFER POINTERS
Ø376				* SOFTWAR	E UART	TO L1NE PRINTER	
Ø377	A2BF	34 17		LA2BF	PSHS	X,B,A,CC	SAVE REGISTERS AND INTERRUPT STATUS
Ø378 Ø379	A2C1	1A 5Ø	22	1 4 2 6 2	ORCC	#\$50	DISABLE IRQ,FIRQ
Ø379 Ø38Ø	A2C3 A2C6	F6 FF 54	22	LA2C3	LDB LSRB	PIA1+2	GET RS 232 STATUS SHIFT RS 232 STATUS BIT INTO CARRY
Ø381	A2C7	25 FA			BCS	LA2C3	LOOP UNTIL READY
Ø382	A2C9	8D 3Ø			BSR	LA2FB	SET OUTPUT TO MARKING
Ø383 Ø384	A2CB A2CC	5F 8D 2F			CLRB BSR	LA2FD	* * TRANSMIT ONE START BIT
Ø385	A2CE	C6 Ø8			LDB	#8	SEND 8 BITS
Ø386	A2DØ	34 Ø4		LA2DØ	PSHS	В	SAVE BIT COUNTER
Ø387	A2D2	5 F			CLRB		CLEAR DA IMAGE I ZEROES TO DA WHEN SENDING RS 232 DATA
Ø388 Ø389	A2D3 A2D4	44 59			LSRA ROLB		ROTATE NEXT BIT OF OUTPUT CHARACTER TO CARRY FLAG * ROTATE CARRY FLAG INTO BIT ONE
Ø39Ø	A2D4	58			ASLB		* AND ALL OTHER BITS SET TO ZERO
Ø391	A2D6	8D 25			BSR	LA2FD	TRANSMIT DATA BYTE
Ø392	A2D8	35 Ø4			PULS	В	GET BIT COUNTER
Ø393 Ø394	A2DA A2DB	5A 26 F3			DECB BNE	LA2DØ	SENT ALL 8 BITS?
Ø395	A2DD	8D 1C			BSR	LA2FB	SEND STOP BIT (ACCB:0)
Ø396	A2DF	35 Ø3			PULS	CC,A	RESTORE OUTPUT CHARACTER & INTERRUPT STATUS
Ø397	A2E1	81 ØD			CMPA	#CR	IS IT CARRIAGE RETURN?
Ø398 Ø399	A2E3 A2E5	27 Ø8 ØC 9C			BEQ INC	LAZED LPTPOS	YES INCREMENT CHARACTER COUNTER
0399 0400	AZES AZES	D6 9C			LDB	LPTPOS	CHECK FOR END OF LINE PRINTER LINE
0401	A2E9	D1 9B			CMPB	LPTWID	AT END OF LINE PRINTER LINE?
0402	A2EB	25 Ø6			BLO	LA2F3	NO
0403	A2ED	ØF 9C		LA2ED	CLR	LPTPOS	RESET CHARACTER COUNTER *
Ø4Ø4 Ø4Ø5	A2EF A2F1	8D 14 8D 12			BSR BSR	LA305 LA305	* DELAY FOR CARRIAGE RETURN
0406	A2F3	F6 FF	22	LA2F3	LDB	PIA1+2	WAIT FOR HANDSHAKE
0407	A2F6	54			LSRB		CHECK FOR R5232 STATUS?
0408	A2F7	25 FA			BCS	LA2F3	NOT YET READY
Ø4Ø9 Ø41Ø	A2F9 A2FB	35 94 C6 Ø2		LA2FB	PULS LDB	B,X,PC #2	RESTORE REGISTERS SET RS232 OUTPUT HIGH (MARKING)
0411	A2FD	F7 FF	20	LA2FD	STB	DA	STORE TO THE D/A CONVERTER REGISTER
0412	A3ØØ	8D ØØ			BSR	LA3Ø2	GO WAIT A WHILE
0413	A3Ø2	9E 95		LA3Ø2	LDX	LPTBTD	GET BAUD RATE
Ø414 Ø415	A3Ø4 A3Ø5	8C 9E 9E 97	97	LA3Ø5	FCB LDX	SKP2 LPTLND	SKIP NEXT TWO BYTES PRINTER CARRIAGE RETURN DELAY
Ø416	A3Ø7	7E A7	D3	271020	JMP	LA7D3	DELAY ON DECREMENTING X
0417							
Ø418	4204	24 16				ER ON THE SCREEN	CAVE DECICTEDS
Ø419 Ø42Ø		34 16 9E 88		LA3ØA	LDX	X,B,A CURPOS	SAVE REGISTERS POINT X TO CURRENT CHARACTER POSITION
0421		81 Ø8			CMPA		IS IT BACKSPACE?
0422		26 ØB			BNE	LA31D	NO
0423		8C Ø4	00			#VIDRAM	AT TOP OF SCREEN?
Ø424 Ø425		27 46 86 6Ø			BEQ LDA	LA35D #\$60	YES - DO NOT ALLOW BACKSPACE BLANK
Ø426		A7 82			STA	, - X	PUT IN PREVIOUS POSITION
Ø427		20 27			BRA	LA344	SAVE NEW CURPOS
Ø428		81 ØD		LA31D	CMPA		ENTER KEY?
Ø429 Ø43Ø		26 ØE 9E 88			BNE LDX	LA32F CURPOS	BRANCH IF NOT GET CURRENT CHAR POSITION
Ø431		86 6Ø		LA323	LDA	#\$60	BLANK
Ø432	A325	A7 8Ø			STA	, χ+	PUT IT ON SCREEN
Ø433		1F 1Ø			TFR	X,D	*
Ø434 Ø435		C5 1F 26 F6			BITB BNE	#\$1F LA323	* TEST FOR BEGINNING OF NEW LINE PUT OUT BLANKS TILL NEW LINE
Ø436		20 15			BRA	LA344	CHECK FOR SCROLLING
Ø437	A32F	81 20		LA32F		#SPACE	*
Ø438		25 2A			BCS	LA35D	* BRANCH IF CONTROL CHARACTER
Ø439 Ø44Ø	A333	4D 2B ØC			TSTA BMI	LA342	SET FLAGS IT IS GRAPHIC CHARACTER
Ø440 Ø441		81 4Ø			CMPA		*
0442		25 Ø6			BCS	LA340	* BRANCH IF NUMBER OR SPECIAL CHARACTER
Ø443		81 60			CMPA		UPPER/LOWER CASE?
Ø444 Ø445		25 Ø4 84 DF			BCS ANDA	LA342 #\$DF	BRANCH IF UPPER CASE ALPHA CLEAR BIT 5, FORCE ASCII LOWER CASE TO BE UPPER CASE
2.45		J. DI					The state of the s

Ø446	A34Ø	88 4	а	LA34Ø E	EORA	#\$40	INVERT BIT 6, CHANGE UPPER CASE TO LOWER & VICE VERSA
Ø447	A342	A7 8			STA	, X+	STORE CHARACTER TO SCREEN
Ø448	A344	9F 8	8		STX	CURPOS	SAVE CURRENT CHAR POSITION
Ø449	A346	8C Ø				#VIDRAM+511	END OF SCREEN BUFFER?
Ø45Ø Ø451	A349 A34B	23 1 8E Ø				LA35D #VIDRAM	RETURN IF NO NEED TO SCROLL TOP OF SCREEN
Ø451	A34B	OL W	שש 14	ı	LUX	#VIDRAM	TOP OF SCREEN
Ø453				* SCROLL TH	HE SCRI	EEN	
Ø454	A34E	EC 8	8 20	LA34E I	LDD	32,X	GET TWO BYTES
0455	A351	ED 8			STD	, X++	MOVE THEM UP ONE ROW
Ø456	A353	8C Ø				#VIDRAM+\$1EØ	AT THE LAST LINE?
Ø457 Ø458	A356 A358	25 F C6 6				LA34E #\$60	NO BLANK
Ø459	A35A	BD A	-			LA92D	BLANK LAST LINE
0460	A35D	35 9	6	LA35D F	PULS	A,B,X,PC	RESTORE REGISTERS
0461							
Ø462						LD WIDTH, TAB ZONE, CURREN ACCORDING TO THE DEVICE S	
Ø463 Ø464	A35F	BD Ø	1 64		JSR	RVEC2	HOOK INTO RAM
Ø465	A362	34 1				X , B , A	SAVE REGISTERS
Ø466	A364	ØF 6	E	(	CLR	PRTDEV	RESET PRINT DEVICE NUMBER
0467	A366	96 6				DEVNUM	GET DEVICE NUMBER
Ø468	A368	27 Ø	9			LA373	BRANCH IF SCREEN
Ø469 Ø47Ø	A36A A36B	4C 27 1	7		INCA BEQ	LA384	CHECK FOR CASSETTE BRANCH IF CASSETTE
Ø471	ASOB	۷, ۱	,	* END UP HE			BRANGII II GASSETTE
0472	A36D	9E 9	9			LPTCFW	TAB FIELD WIDTH AND TAB ZONE
0473	A36F	DC 9	-			LPTWID	PRINTER WIDTH AND POSITION
0474	A371	2Ø Ø	9			LA37C	SET PRINT PARAMETERS
Ø475 Ø476	A373	D6 8	0	* SCREEN DI LA373 L		VALUES CURPOS+1	GET CURSOR LOC LS BYTE
Ø470	A375	C4 1				#\$1F	KEEP ONLY COLUMN POSITION
Ø478	A377	8E 1				#\$1010	TAB FIELD WIDTH AND LAST TAB ZONE
Ø479	A37A	86 2	Ø	l	LDA	#32	DISPLAY SCREEN LINE WIDTH
0480	A37C	9F 6				DEVCFW	SAVE TAB FIELD WIDTH AND ZONE
Ø481	A37E	D7 6				DEVPOS	SAVE PRINT POSITION
Ø482 Ø483	A38Ø A382	97 6 35 9				DEVWID A,B,X,PC	SAVE PRINT WIDTH RESTORE REGISTERS
Ø484	A384	Ø3 6	-		COM	PRTDEV	SET TO \$FF FOR CASSETTE
Ø485	A386	8E Ø	1 00	l	LDX	#\$0100	* TAB FIELD WIDTH = 1; ALL OTHER
Ø486	A389	4 F			CLRA		* PARAMETERS = Ø
Ø487	A38A	5 F	_		CLRB		*
Ø488 Ø489	A38B	2Ø E	F	Ŀ	BRA	LA37C	SET PRINT PARAMETERS
0490				* THIS IS T	THE ROI	JTINE THAT GETS AN INPUT L	INE FOR BASIC
0491				* EXIT WITH	H BREA	K KEY: CARRY = 1	
Ø492				* EXIT WITH	H ENTE	R KEY: CARRY = Ø	
Ø492 Ø493	A38D	BD A		* EXIT WITH LA38D &	H ENTEI JSR	R KEY: CARRY = Ø LA928	CLEAR SCREEN
Ø492 Ø493 Ø494	A39Ø	BD Ø	1 82	* EXIT WITH LA38D C LA39Ø C	H ENTEI JSR JSR	R KEY: CARRY = Ø LA928 RVEC12	HOOK INTO RAM
Ø492 Ø493	A39Ø A393	BD Ø ØF 8	1 82 7	* EXIT WITH LA38D & LA39Ø &	H ENTEI JSR JSR CLR	R KEY: CARRY = Ø LA928	
Ø492 Ø493 Ø494 Ø495	A39Ø	BD Ø	1 82 7 2 DD	* EXIT WITH LA38D & LA39Ø & (	H ENTEI JSR JSR CLR LDX	R KEY: CARRY = Ø LA928 RVEC12 IKEYIM	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE
Ø492 Ø493 Ø494 Ø495 Ø496 Ø497 Ø498	A39Ø A393 A395 A398	BD Ø ØF 8 8E Ø C6 Ø	1 82 7 2 DD 1	* EXIT WITH LA38D & LA390 & ( L	H ENTEI JSR JSR CLR LDX LDB	R KEY: CARRY = Ø LA928 RVEC12 IKEYIM #LINBUF+1 #1	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER
Ø492 Ø493 Ø494 Ø495 Ø496 Ø497 Ø498 Ø499	A39Ø A393 A395 A398	BD Ø ØF 8 8E Ø C6 Ø	1 82 7 2 DD 1 1 71	* EXIT WITH LA380	H ENTEI JSR JSR CLR LDX LDB JSR	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN
0492 0493 0494 0495 0496 0497 0498 0499 0500	A39Ø A393 A395 A398 A39A A39D	BD Ø ØF 8 8E Ø C6 Ø BD A ØD 7	1 82 7 2 DD 1 1 71 Ø	* EXIT WITH LA38D	H ENTEI JSR JSR CLR LDX LDB JSR TST	R KEY: CARRY = Ø LA928 RVEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG
0492 0493 0494 0495 0496 0497 0498 0499 0500 0501	A39Ø A393 A395 A398 A39A A39D A39F	BD Ø ØF 8 8E Ø C6 Ø BD A ØD 7 26 2	1 82 7 2 DD 1 71 Ø B	* EXIT WITH LA38D 6 LA39Ø 6 L  * LA39A 6 LA39A 6	H ENTEI JSR JSR CLR LDX LDB JSR TST BNE	R KEY: CARRY = Ø LA928 RVEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE
0492 0493 0494 0495 0496 0497 0498 0499 0500	A39Ø A393 A395 A398 A39A A39D	BD Ø ØF 8 8E Ø C6 Ø BD A ØD 7	1 82 7 2 DD 1 71 Ø B	* EXIT WITH LA380	H ENTER JSR JSR CLR LDX LDB JSR TST BNE TST	R KEY: CARRY = Ø LA928 RVEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG
0492 0493 0494 0495 0496 0497 0498 0499 0500 0501 0502 0503 0504	A39Ø A393 A395 A398 A39A A39D A39F A3A1	BD Ø ØF 8 8E Ø C6 Ø Ø P 7 26 2 Ø D 6 26 2 81 Ø	1 82 7 2 DD 1 71 Ø B F 3 C	* EXIT WITH LA38D LA39Ø ( LA39Ø ( LA39A LA39A LA39A LA39A LA39A	H ENTER JSR JSR CLR LDX LDB JSR TST BNE TST BNE	R KEY: CARRY = Ø LA928 RVEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC DEVNUM	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER
0492 0493 0494 0495 0496 0497 0498 0499 0500 0501 0502 0503 0504 0505	A390 A393 A395 A398 A39A A39A A39F A3A1 A3A3 A3A5	BD Ø Ø Ø F 8 8 E Ø C6 Ø Ø BD A Ø D 7 26 2 Ø D 6 26 2 81 Ø 27 E	1 82 7 2 DD 1 71 Ø B F 3 C 4	* EXIT WITH LA38D	H ENTEI JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BEQ	R KEY: CARRY = 0 LA928 RVEC12 IKEYIM #LINBUF+1 #1 LA171 CLINBFL LA3CC DEVNUM LA3CB #FORMF	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCE CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN
0492 0493 0494 0495 0497 0498 0499 0500 0501 0502 0503 0504 0505	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A3 A3A5 A3A7 A3A9	BD Ø ØF 8 8E Ø C6 Ø Ø P P P P P P P P P P P P P P P P P	1 82 7 2 DD 1 71 Ø B F 3 C 4 8	* EXIT WITH LA38D LA390 (  L  * LA39A  LA39A  E  C  C  C  C  C  C  C  C  C  C  C  C	H ENTEL JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BEQ CMPA	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #BS	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE
0492 0493 0494 0495 0496 0497 0498 0499 0500 0501 0502 0503 0504 0505	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A3 A3A5 A3A7 A3A9	BD Ø ØF 8 8E Ø C6 Ø Ø BD A ØD 7 26 2 ØD 6 26 2 81 Ø 27 E 81 Ø 26 Ø	1 82 7 2 DD 1 71 Ø B F 3 C 4 8	* EXIT WITH LA380 LA390 (	H ENTEL JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BEQ CMPA	R KEY: CARRY = 0 LA928 RVEC12 IKEYIM #LINBUF+1 #1 LA171 CLINBFL LA3CC DEVNUM LA3CB #FORMF	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCE CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0504 0505 0506 0507 0508	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A3 A3A5 A3A7 A3A9 A3AB	BD Ø Ø F 8 8 E Ø C6 Ø Ø F 8 8 E Ø Ø C6 Ø Ø F 8 E Ø Ø E Ø E Ø E Ø E Ø E Ø E Ø E Ø E Ø	1 82 7 2 DD 1 71 Ø B F 3 C 4 8 8	* EXIT WITH LA38D LA390 C LA390 C LA39A C LA39A C E C C E C C E C C E C C E C C E C C E C E C E C E C E C E C E C E C E E C E C E E C E E C E E E C E E E E E E E E E E E E E E E E E E E E	H ENTEI JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BEQ CMPA BNE	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171 CCINBFL LA3CC DEVNUM LA3CB #FORMF LA38D	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO
0492 0493 0494 0495 0496 0497 0498 0500 0501 0502 0503 0504 0505 0506 0507 0506 0507 0509 0510	A390 A393 A395 A398 A39A A39F A3A1 A3A3 A3A5 A3A7 A3A9 A3AB A3AB A3AB	BD Ø ØF 8 8E Ø C6 Ø Ø BD A ØD 7 26 2 ØD 6 26 2 81 Ø 26 Ø 26 Ø 27 E 81 Ø 26 Ø Ø 5A 27 E 3Ø 1	1 82 7 2 DD 1 71 0 B F 3 C 4 4 8 7	* EXIT WITH LA380	H ENTEL JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BEQ CMPA BNE DECB	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #BS LA3B4 LA39Ø	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER
0492 0493 0494 0495 0496 0497 0498 0500 0501 0502 0503 0504 0505 0506 0507 0508 0509 0510 0511	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A3 A3A5 A3A7 A3A9 A3AB A3AB A3AB A3B0 A3B2	BD ØF 88E ØC6 Ø  BD A ØD 7 26 2 ØD 66 26 2 81 Ø 27 E 81 Ø 26 Ø 27 E 30 1 20 3	1 82 7 7 2 DD 1 1 71 8 8 8 7 7 8 8 8 7 7 8 8 7 7	* EXIT WITH LA38D LA390  (  LA39A  LA39A  E  C  E  C  E  C  E  C  E  E  C  E  E	H ENTEL JSR JSR CLR LDX LDB JSR TST BNE CMPA BEQ CMPA BNE	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #85 LA38D #85 LA384 LA390 -1,X LA3E8	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN
0492 0493 0494 0495 0496 0497 0498 0500 0501 0502 0503 0504 0505 0506 0507 0508 0509 0511 0512	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A5 A3A7 A3A9 A3AB A3AB A3AB A3AB A3AB A3AB A3AB	BD Ø ØF 8 8E Ø C6 Ø Ø P 7 26 2 281 Ø 27 E 81 Ø 26 Ø 5A 27 E 30 1 20 3 81 1	1 82 7 2 DD 1 1 71 9 B F 3 3 C C 4 4 8 8 7 9 9 F 5 4 5 5	* EXIT WITH LA380	H ENTEL JSR JSR CLR LDX LDB JSR TST BNE CMPA BEQ CMPA BNE DECB BEQ LEAX BRA CMPA	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #8S LA38B LA384 LA39Ø -1,X LA39Ø +1,X LA328 #\$15	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW?
0492 0493 0494 0495 0496 0497 0498 0499 0500 0500 0500 0500 0500 0506 0507 0508 0508 0508 0508 0508 0508 0508	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A3 A3A5 A3A7 A3A9 A3AB A3AB A3AB A3B0 A3B2	BD Ø ØF 8 8E Ø C6 Ø Ø P 7 26 2 281 Ø 27 E 81 Ø 26 Ø 5A 27 E 30 1 20 3 81 1	1 82 7 2 DD 1 1 71 9 B F 3 3 C C 4 4 8 8 7 9 9 F 5 4 5 5	* EXIT WITH LA38D LA390  (  LA39A  LA39A  E  C  E  C  E  C  LA38B4  E  LA3B4	H ENTEI JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BNE DECB BEQ LEAX BRA BRA CMPA BRA BRA	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA3CB #BS LA3B4  LA390 -1,X LA3EB #\$15 LA3C2	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FFED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO
0492 0493 0494 0495 0496 0497 0498 0499 0500 0501 0502 0503 0504 0505 0506 0507 0508 0507 0508 0509 0510 0511 0512 0512	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A3 A3A5 A3A7 A3A9 A3AB A3AB A3AB A3B0 A3B2 A3B6	BD Ø ØF 8 8E Ø C6 Ø Ø 6 26 2 81 Ø 26 Ø 5A 27 E 3Ø 1 26 Ø 3 81 1 26 Ø	1 82 7 2 DD 1 1 71 9 B F 3 3 C C 4 4 8 8 7 9 9 F 5 4 5 5	* EXIT WITH LA38D LA390   * LA39A  E E E E E LA3B4 E * YES, RESE	H ENTEI JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BNE BNE CMPA BNE BNE BNE CMPA BNE BNE CMPA BNE BNE CMPA CMPA CMPA CMPA CMPA CMPA CMPA CMPA	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #8S LA38B LA384 LA39Ø -1,X LA39Ø +1,X LA328 #\$15	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FFED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0504 0505 0506 0507 0508 0509 0510 0511 0512 0513 0513	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A5 A3A7 A3A9 A3AB A3AB A3AB A3AB A3AB A3AB A3AB	BD Ø ØF 8 8E Ø C6 Ø Ø BD A ØD 7 26 2 ØD 6 26 27 E 81 Ø 26 Ø 5A 27 E 3Ø 1 2Ø 3 81 1 26 Ø 5A	1 82 7 DD 1 1 71 1 71 1 71 8 B F 3 C C 4 4 8 8 7 Ø F F 4 4 5 5 A	* EXIT WITH LA38D LA390	H ENTEI JSR JSR CLR LDX LDB JSR TST BNE TST BNE CMPA BNE DECB BEQ LEAX BRA BRA CMPA BRA BRA	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #85 LA384 LA390 -1,X LA3E8 #\$15 LA3C2 FER TO BEGINNING AND ERASE	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE
0492 0493 0494 0495 0496 0497 0498 0490 0501 0502 0503 0504 0505 0506 0507 0508 0507 0508 0510 0511 0512 0513 0514 0515 0516	A390 A393 A395 A398 A399 A397 A3A1 A3A3 A3A5 A3A7 A3A9 A3AB A3AB A3B0 A3B2 A3B4 A3B6 A3B8 A3B8 A3B8 A3B8 A3B8	BD 0 0 F 8 8 E 0 0 0 F 8 8 E 0 0 0 F 8 8 E 0 0 0 F 8 E 0 E 0 E 0 E 0 E 0 E 0 E 0 E 0 E 0 E	1 82 7 2 DD 1 1 71 1 71 0 8 8 7 0 7 7 4 4 5 5 A 5 5 8	* EXIT WITH LA38D LA390  * LA39A  * LA39A  E E E  LA3B4  * YES, RESE LA3B8	H ENTEI JSR JSR JCR JCR JCR JCR JCR JCR JCR JCR JCR JC	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1 LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA3B4 LA3B4 LA3B4 LA390 -1,X LA3E8 #\$15 LA3C2 FER TO BEGINNING AND ERASE LA390 #BS	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE?
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0504 0505 0506 0507 0511 0512 0511 0512 0514 0515 0516	A390 A393 A395 A398 A39A A39D A39F A3A1 A3A3 A3A7 A3A9 A3AB A3AB A3AB A3AB A3BB A3BB A3BB	BD 0 8 8 8 6 0 9 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9	1 82 7 2 DD 1 1 71 1 71 0 0 8 8 7 7 0 0 F 4 4 5 5 8 8 2 82	* EXIT WITH LA38D LA390  * LA39A  E E E E E LA384  * YES, RESE LA388  E	H ENTEI JSR JSR LCLR LDX LLDB JSR TST ST	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #BS LA384  LA390 -1, X LA3E8 #515 LA3C2 FER TO BEGINNING AND ERASE LA390 #BS PUTCHR	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN)
0492 0493 0494 0495 0496 0497 0498 0499 0500 0502 0503 0506 0507 0508 0508 0510 0511 0512 0513 0514 0515 0516 0517 0518	A398 A395 A398 A396 A397 A397 A341 A3A3 A3A5 A3A7 A3A9 A3A8 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8	BD 0 0 8 8 8 0 0 0 6 6 0 0 0 0 0 0 0 0 0 0	1 82 7 2 DD 1 1 71 1 71 0 8 8 7 0 9 F 4 4 5 5 A 5 5 8 8 2 82 6	* EXIT WITH LA38D LA390	H ENTEI JSR JSR CLR LDX LDX LDB JSR TST TST TST TST TST ENE CMPA BBEQ CMPA BBEQ LEAX MMPA BBEQ LEAX BBEQ LEAX BBEQ LEAX LEAX BBEQ LEAX LEAX BBEQ LEAX BBEQ LEAX BBEQ LLEAX BBEQ LLEAX BBEQ LLEAX BBEQ LLEAX BBEQ LLEAX BBEQ LLEAX BBEA BBEA BBEA BBEA BBEA BBEA BBEA B	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #BS LA3B4 LA390 -1, X LA3E2 FER TO BEGINNING AND ERASE LA390 #BS LA380	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING
0492 0493 0494 0495 0496 0497 0498 0500 0500 0500 0500 0500 0507 0508 0506 0507 0508 0507 0510 0511 0512 0513 0514 0515 0516 0517 0518 0519	A390 A393 A398 A398 A397 A341 A343 A345 A346 A346 A346 A346 A346 A346 A346 A346	BD 0 0 7 8 8 8 6 0 0 6 6 9 8 9 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9	1 82 7 2 DD 1 1 71 1 71 8 B F F 3 C C 4 4 8 8 7 8 9 F F 4 5 5 A 5 5 8 2 8 2 8 2 6 6 3	* EXIT WITH LA38D LA390 C LA390 C L T L T LA39A C E C C E C C E C C E C C E C C E C C E C C E C C E C C E C C E C C E C C E C C E C C E C C C E C C C E C C C E C C C E C C C E C C C C E C C C C E C C C C C C C C C C C C C C C C C C C C	H ENTEI JSR JSR LCLR LDX LLDB JSR TST ST	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA380 #BS LA384  LA390 -1,X LA3E8 #515 LA3C2 FER TO BEGINNING AND ERASE  LA390 #BS LA390 #BS LA388 #BS LA388 #BS LA388 #BS LA388 #BS LA388	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING KEEP GOING BREAK KEY?
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0506 0507 0508 0508 0511 0512 0511 0512 0514 0515 0516 0517 0518	A398 A395 A398 A396 A397 A397 A341 A3A3 A3A5 A3A7 A3A9 A3A8 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8	BD 0 0 8 8 8 6 6 6 0 0 0 0 0 0 0 0 0 0 0 0	1 82 77 22 DD 1 1 71 71 88 8 7 8 8 8 7 8 8 8 2 8 2 8 2 8 2 8 3 1 1	* EXIT WITH LA38D LA390	H ENTEI JSR JSR LOX LDX LDB LDX LDB LDX LDB	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #BS LA384  LA390 -1, X LA3E8 #515 LA3C2 FER TO BEGINNING AND ERASE LA390 #BS PUTCHR LA388 #3 #1	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING
0492 0493 0494 0495 0496 0497 0498 0499 0500 0500 0500 0500 0507 0508 0507 0510 0512 0513 0514 0515 0516 0517 0516 0517 0519 0519 0519 0519	A398 A398 A398 A398 A397 A341 A343 A345 A346 A346 A348 A348 A348 A348 A348 A348 A348 A348	BD 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 82 7 7 2 DD 1 1 71 1 71 1 71 8 8 8 7 7 9 F 4 5 5 A 5 8 8 2 8 2 6 6 3 3 1 5 5 D	* EXIT WITH LA380 LA390	H ENTEI JSR JSR CLR LDX LDX LDB JSR TST SENE LTST SENE LTST SENE LTST SENE LEAX BEQ LEAX BEQ LEAX BEQ LEAX BEQ LEAX LEAX BEA BEQ LEAX BEA BEQ LEAX CMPA BEQ LEAX CMPA BEQ CMPA BEQ CMPA CMPA CMPA CMPA CMPA CMPA CMPA CMPA	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38B #S LA3B4  LA390 -1, X LA3E8 #515 LA3C2 FER TO BEGINNING AND ERASE LA390 #85 PUTCHR LA388 #3 #1 LA3CD #CR	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY?
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0504 0505 0506 0506 0511 0512 0511 0512 0513 0514 0515 0516 0517 0518 0519 0519 0519 0519	A398 A398 A398 A398 A398 A397 A3A1 A3A5 A3A7 A3A8 A3A8 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8	BD	1 82 7 7 2 DD 1 1 71 1 71 1 71 8 8 8 7 7 9 F 4 5 5 A 5 8 8 2 8 2 6 6 3 3 1 5 5 D	* EXIT WITH LA38D LA390	H ENTEI JSR JSR LDX LDX LDX LDB JSR TST TST TST TST TST TST TST TST TST T	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38B #S LA3B4  LA390 -1, X LA3E8 #515 LA3C2 FER TO BEGINNING AND ERASE LA390 #85 PUTCHR LA388 #3 #1 LA3CD #CR	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0506 0507 0508 0508 0509 0511 0512 0513 0515 0516 0516 0517 0518 0519 0522 0523 0522 0523	A398 A398 A398 A398 A398 A399 A397 A3A1 A3A3 A3A6 A3A6 A3B4 A3B6 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8 A3B8	BD 0 0 8 8 8 6 6 0 0 0 0 7 2 6 2 8 1 0 0 1 2 7 8 8 1 1 2 6 0 0 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7 1 7	1 82 7 7 2 DD 1 1 71 8 8 7 7 8 8 7 7 8 8 8 2 8 2 8 2 8 2 8 2	* EXIT WITH LA38D LA390	H ENTEI JSR JSR JSR LDX LDX LDX LDB JSR TENE TENE TENE TEST ENDE ENDE ENDE ENDE ENDE ENDE ENDE END	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #85 LA384 LA390 -1,X LA3E8 #\$15 LA3C2 FER TO BEGINNING AND ERASE LA390 #85 PUTCHR LA388 #81 LA398 #85 LA3B8 #87 H1 LA3CD #CR LA3CD	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CUURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEFF GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG IF ENTER KEY - END LINE ENTRY
0492 0493 0494 0495 0496 0497 0498 0499 0500 0502 0503 0506 0507 0508 0510 0511 0512 0513 0514 0515 0516 0517 0518 0519 0519 0520 0522 0523 0522 0523	A398 A398 A398 A398 A398 A397 A3A1 A3A3 A3A5 A3A8 A3A8 A3A8 A3B8 A3B8 A3B8 A3B8 A3B8	BD 0 0 F 8 8 C 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 82 7 7 2 DD 1 1 71 9 8 8 8 7 9 8 8 2 8 2 8 2 8 2 8 2 1 5 DD 1	* EXIT WITH LA38D LA390	H ENTEI JSR JSR CLR LDX LDX LDB JSR TST TST TST TST TST ENE CMPA BBEQ CMPA BBEQ LEAX BBEQ LEAX BBEQ LEAX BBEQ LEAX BBEQ CMPA BBEQ CMPA BBEQ CMPA CMPA CMPA CMPA CMPA CMPA CMPA CMPA	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #BS LA3B4 LA390 -1,X LA3E2 FER TO BEGINNING AND ERASE LA326 #BS #BS LA328 #11 LA3CD #CR LA300 #CR LA300 #CC	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG BRANCH IF BREAK KEY - END LINE ENTRY SAVE CARRY FLAG
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0506 0507 0508 0508 0509 0511 0512 0513 0515 0516 0516 0517 0518 0519 0522 0523 0522 0523	A398 A398 A398 A398 A398 A397 A341 A343 A345 A346 A348 A348 A348 A348 A348 A348 A348 A348	BD 0 0 8 8 8 6 6 6 0 0 0 0 0 0 0 0 0 0 0 0	1 82 7 7 2 DD 1 1 71 1 71 8 8 8 8 7 8 7 8 8 8 8 8 8 2 8 2 8 2 8 2	* EXIT WITH LA38D LA390   * LA39A  LA39A  E E C E LA3B4  * YES, RESE LA3B8  LA3C2  LA3C2  LA3C2  LA3CC LA3CD  LA3CC LA3CD	H ENTEI JSR JSR JSR LDX LDX LDX LDB JSR TENE TENE TENE TEST ENDE ENDE ENDE ENDE ENDE ENDE ENDE END	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #85 LA38D #85 LA38B #15 LA32C EFR TO BEGINNING AND ERASE LA390 #85 PUTCHR LA38B #81 LA390 #85 PUTCHR LA300 #6CR LA309 CC L8958	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CUURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEFF GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG IF ENTER KEY - END LINE ENTRY
0492 0493 0494 0495 0496 0497 0498 0490 0501 0502 0503 0504 0505 0506 0507 0508 0507 0508 0509 0511 0512 0513 0514 0515 0516 0517 0518 0517 0518 0519 052 052 052 052 052 052 052 052 052 052	A398 A398 A398 A398 A398 A397 A347 A347 A347 A348 A348 A348 A386 A386 A386 A386 A386 A386 A386 A38	BD 0 0 8 8 8 8 6 6 6 9 0 0 7 2 6 2 8 1 0 0 6 8 8 8 1 0 2 7 E E 8 1 0 0 2 7 E E 8 1 0 0 2 7 E E 8 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	1 82 7 7 DD 1 7 1 7 1 9 8 2 8 2 8 2 8 2 8 2 8 2 8 2 8 3 1 5 5 DD 1 9 5 8 4	* EXIT WITH LA38D LA390	H ENTEI JSR JSR SISR CCLR LDX LDX LDB JSR TEST TEST TEST ENDE TST ENDE ENDE ENDE ENDE ENDE ENDE ENDE END	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #85 LA38D #85 LA38B #15 LA32C EFR TO BEGINNING AND ERASE LA390 #85 PUTCHR LA38B #81 LA390 #85 PUTCHR LA300 #6CR LA309 CC L8958	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG SEND CR TO SCREEN
0492 0493 0494 0495 0496 0497 0498 0490 0501 0502 0503 0504 0505 0506 0507 0508 0509 0511 0512 0513 0514 0515 0516 0517 0518 0517 0518 0519 052 052 052 052 052 052 052 052 052 052	A398 A398 A398 A398 A398 A397 A341 A343 A345 A346 A348 A348 A348 A348 A348 A348 A348 A348	BD 0 0 8 8 8 8 6 6 6 9 0 0 7 2 6 2 8 1 0 0 6 8 8 8 1 0 2 7 E E 8 1 0 0 2 7 E E 8 1 0 0 2 7 E E 8 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	1 82 7 7 DD 1 71 1 0 8 F 7 8 2 8 2 8 2 8 2 8 2 BD DD 1 9 5 8 4 2 DC	* EXIT WITH LA38D LA390	H ENTEI JSR JSR JSR CLR LDX LDX LDB JSR TST TST TST TST TST ENE CMPA BBEQ CMPA BBEQ LEAX BBEQ LEAX BBEQ LEAX BBEQ LEAX BBEQ CMPA BBEQ LEAX BBEQ CMPA BBEQ LEAX BBEQ LEAX CMPA BBEQ LEAX CMPA BBEQ LEAX CMPA BBEQ LEAX CMPA BBEQ LCAX CMPA BBEQ LCAX CMPA BBC CMPA CMPA CMPA CMPA CMPA CMPA CMPA CMP	R KEY: CARRY = 0 LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #8S LA384  LA390 -1, X LA3E8 #\$15 LA3C2 FER TO BEGINNING AND ERASE LA398 #\$15 LA3C3 #\$15 LA3C4 LA398 #\$15 LA3C5 CCC LEVENT AND LA3C5 #CCC LEP558 , X	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG BRANCH IF BNEAK KEY DOWN ENTER KEY? SAVE CARRY FLAG SEND CR TO SCREEN MAKE LAST BYTE IN INPUT BUFFER = Ø
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0506 0507 0508 0507 0511 0512 0511 0512 0511 0512 0511 0512 0511 0512 0511 0512 0514 0515 0516 052 052 052 052 052 052 052 052 052 053 053 053 054 054 055 055 056 056 056 056 056 056 056 056	A398 A398 A398 A398 A398 A397 A341 A343 A345 A346 A348 A348 A348 A348 A348 A348 A348 A348	BD 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 82 7 7 DD 1 71 1 0 8 F 7 8 2 8 2 8 2 8 2 8 2 BD DD 1 9 5 8 4 2 DC	* EXIT WITH LA38D LA390	H ENTEI JOSR JOSR LDX LDX LDX LDX LDB  SR TEST TEST TEST TEST TEST TEST TEST	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3C8 #FORMF LA38D #BS LA384  LA390 -1, X LA3E8 #515 LA3C2 FER TO BEGINNING AND ERASE LA398 #81 LA390 #85 PUTCHR LA388 #3 #1 LA3CO #CC L8958 , X #LINBUF CC, PC	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG IF ENTER KEY - END LINE ENTRY SAVE CARRY FLAG SEND CR TO SCREEN MAKE LAST BYTE IN INPUT BUFFER = Ø RESET INPUT BUFFER POINTER RESTORE CARRY FLAG
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0506 0507 0508 0508 0509 0510 0511 0512 0513 0515 0516 0517 0518 0519 0522 0523 0522 0523 0526 0527 0528 0529 0529 0529 0529	A398 A398 A398 A398 A398 A397 A343 A345 A346 A346 A348 A348 A348 A348 A348 A348 A348 A348	BD 0 0 F 8 8 C 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 82 7 7 DD 1 71 1 0 BF 3 CC 4 8 8 7 0 FF 4 5 A 5 8 2 6 6 3 1 5 DD D 1 5 8 4 2 1 1 5 DD D 1 1 9 4 2 1 1	* EXIT WITH LA38D LA390	H ENTEI JSR JSR SS S	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #8S LA3B4 LA39Ø -1,X LA3E8 #\$15 LA3C2 FER TO BEGINNING AND ERASE LA3B8 #85 LA3B4 LA39Ø #85 LA3B4 LA39Ø #85 LA3C2 CEURINE LA3C2 FER TO BEGINNING AND ERASE LA3B8 #3 #1 LA3CD #CC LB958 ,X #LINBUF CC, PC CTER INTO THE BASIC LINE I	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER  ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEFP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG SEND CR TO SCREEN MAKE LAST BYTE IN INPUT BUFFER = Ø RESET INPUT BUFFER POINTER RESTORE CARRY FLAG
0492 0493 0494 0495 0496 0497 0498 0499 0501 0502 0503 0506 0507 0508 0508 0509 0510 0511 0512 0513 0515 0516 0517 0518 0519 0522 0523 0522 0523 0526 0527 0528 0529 0529 0529 0529	A398 A398 A398 A398 A398 A397 A341 A343 A345 A346 A348 A348 A348 A348 A348 A348 A348 A348	BD 0 0 F 8 8 C 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 82 77 DD 1 71 1 71 1 71 1 71 1 71 1 71 1	* EXIT WITH LA38D LA390 C I I X LA39A C E C E C E C E C E C E C E C E C E C	H ENTEI JSR JSR CLR LDX LDB JSR TST BNE TST BNE TST BNE CMPA BEQ CMPA BEQ DECB BEQ CMPA BBR BNE BBR BNE BBR BBR BBR CMPA CMPA DECB BBR BC CLR BBR BC CLR CMPA CLR CLR CLR CLR CLR CMPA CLR CLR CLR CMPA CLR CMPA CLR CMPA CMPA CMPA CMPA CMPA CMPA CMPA CMPA	R KEY: CARRY = Ø LA928 RYEC12 IKEYIM #LINBUF+1 #1  LA171 CINBFL LA3CC DEVNUM LA3CB #FORMF LA38D #8S LA3B4 LA39Ø -1,X LA3E8 #\$15 LA3C2 FER TO BEGINNING AND ERASE LA3B8 #85 LA3B4 LA39Ø #85 LA3B4 LA39Ø #85 LA3C2 CEURINE LA3C2 FER TO BEGINNING AND ERASE LA3B8 #3 #1 LA3CD #CC LB958 ,X #LINBUF CC, PC CTER INTO THE BASIC LINE I	HOOK INTO RAM RESET BREAK CHECK KEY TEMP KEY STORAGE INPUT LINE BUFFER ACCB CHAR COUNTER: SET TO 1 TO ALLOW A BACKSPACE AS FIRST CHARACTER GO GET A CHARACTER FROM CONSOLE IN GET CONSOLE IN BUFFER FLAG BRANCH IF NO MORE CHARACTERS IN INPUT FILE CHECK DEVICE NUMBER BRANCH IF NOT SCREEN FORM FEED YES - CLEAR SCREEN BACKSPACE NO YES - DECREMENT CHAR COUNTER BRANCH IF BACK AT START OF LINE AGAIN DECREMENT BUFFER POINTER ECHO CHAR TO SCREEN SHIFT RIGHT ARROW? NO CURRENT LINE DEC CHAR CTR GO BACK TO START IF CHAR CTR = Ø BACKSPACE? SEND TO CONSOLE OUT (SCREEN) KEEP GOING BREAK KEY? SET CARRY FLAG BRANCH IF BREAK KEY DOWN ENTER KEY? NO CLEAR CARRY FLAG IF ENTER KEY - END LINE ENTRY SAVE CARRY FLAG SEND CR TO SCREEN MAKE LAST BYTE IN INPUT BUFFER = Ø RESET INPUT BUFFER POINTER RESTORE CARRY FLAG

Ø535							
	A3DD	81 7E			СМРА	#'z-1	*
Ø536	A3DF	24 B9			BCC	# 2-1 LA39A	* IGNORE IF > LOWER CASE Z
Ø537		C1 FA				#LBUFMX	HAVE 250 OR MORE CHARACTERS BEEN ENTERED?
Ø538		24 B5			BCC	LA39A	YES, IGNORE ANY MORE
Ø539		A7 80	)		STA	, X+	PUT IT IN INPUT BUFFER
0540	A3E7				INCB		INCREMENT CHARACTER COUNTER
0541		BD A2		LA3E8	JSR	PUTCHR	ECHO IT TO SCREEN
Ø542	A3EB	20 AD	)		BRA	LA39A	GO SET SOME MORE
Ø543							
Ø544				* INPUT [	DEVICE	NUMBER CHECK	
Ø545	A3ED	BD Ø1	. 6D	LA3ED	JSR	RVEC5	HOOK INTO RAM
Ø546	A3FØ	96 6F	:		LDA	DEVNUM	DEVICE NUMBER
Ø547	A3F2	27 21			BEQ	LA415	RETURN IF SCREEN
Ø548	A3F4	4 C			INCA		*
Ø549	A3F5	26 Ø0			BNE	LA4Ø3	* BRANCH IF NOT CASSETTE (BAD FILE MODE)
0550		96 78			LDA	FILSTA	GET FILE STATUS
Ø551		26 Ø5			BNE	LA400	FILE IS OPEN
	A3FB			LA3FB	LDB	#22*2	FILE NOT OPEN ERROR
Ø553	A3FD			LASTE	JMP	LAC46	JUMP TO ERROR SERVICING ROUTINE
Ø554	A400		, 40	LA4ØØ	DECA	ENCTO	*
Ø555		27 12	,	LATED	BEQ	LA415	* FILE IS IN INPUT MODE, RETURN
				1.4400			
Ø556	A403	7E A6	) 10	LA4Ø3	JMP	LA616	BAD FILE MODE ERROR
Ø557							
Ø558						NUMBER CHECK	HARM THEO PAN
Ø559		BD Ø1		LA4Ø6	JSR	RVEC6	HOOK INTO RAM
0560		96 6F			LDA	DEVNUM	GET DEVICE NUMBER
0561	A4ØB				INCA		*
	A4ØC				BNE	LA415	* RETURN IF NOT TAPE
Ø563	A4ØE	96 78	3		LDA	FILSTA	GET FILE STATUS
Ø564	A410	27 E9	)		BEQ	LA3FB	FILE NOT OPEN ERROR
Ø565	A412	4 A			DECA		*
Ø566	A413	27 EE			BEQ	LA4Ø3	* BAD FILE MODE - FILE IN INPUT MODE
Ø567	A415	39		LA415	RTS		
Ø568							
Ø569				* CLOSE			
Ø57Ø	A416	27 ØE		CLOSE	BEQ	LA426	BRANCH IF NO NAME SPECIFIED
Ø571		BD AS			JSR	LA5A5	CHECK DEVICE NUMBER
Ø572		8D 10		LA41B	BSR	LA42D	GO CLOSE A FILE
Ø572		9D A5		LATID	JSR	GETCCH	GET CURRENT BASIC CHARACTER
Ø573		27 2			BEQ	LA44B	RETURN IF NO MORE FILES
Ø574							CHECK SYNTAX AND DEVICE NUMBER
		BD AS			JSR	LASA2	
Ø576	A424	2Ø F5	•		BRA	LA41B	KEEP CLOSING FILES
Ø577							
Ø578						ES HANDLER	
Ø579		BD Ø1		LA426	JSR	RVEC7	HOOK INTO RAM
Ø58Ø		86 FF			LDA	#-1	CASSETTE DEVICE NUMBER
Ø581	A42B	97 6F	:		STA	DEVNUM	SET DEVICE NUMBER
Ø582				* CLOSE F	ILE HA	NDLER	
Ø583	A42D	BD Ø1	. 76	LA42D	JSR	RVEC8	HOOK INTO RAM
Ø584	A43Ø	96 6F	:		LDA	DEVNUM	GET DEVICE NUMBER
~ - ~ -	A432	ØF 6F			CLR	DEVNUM	SET TO SCREEN
Ø585		4 C			INCA		*
Ø586	A434	26 14	ļ.		BNE	LA44B	* BRANCH IF WAS NOT CASSETTE
					1.04	FILSTA	GET FILE STATUS
Ø586 Ø587	A435		3		LUA		
Ø586 Ø587 Ø588	A435 A437	96 78			LDA CMPA	#2	IS IT OUTPUT MODE
Ø586 Ø587 Ø588 Ø589	A435 A437 A439	96 78 81 Ø2	2		CMPA	#2 LA449	IS IT OUTPUT MODE
Ø586 Ø587 Ø588 Ø589 Ø59Ø	A435 A437 A439 A43B	96 78 81 Ø2 26 Ø0	<u> </u>		CMPA BNE	LA449	NO
Ø586 Ø587 Ø588 Ø589 Ø59Ø	A435 A437 A439 A43B A43D	96 78 81 Ø2 26 Ø0 96 79	2		CMPA BNE LDA	LA449 CINCTR	NO GET CHARACTER BUFFER CTR
0586 0587 0588 0589 0590 0591 0592	A435 A437 A439 A43B A43D A43F	96 78 81 Ø2 26 Ø0 96 79 27 Ø3	<u>?</u> ; )		CMPA BNE LDA BEQ	LA449 CINCTR LA444	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY
Ø586 Ø587 Ø588 Ø589 Ø590 Ø591 Ø592 Ø593	A435 A437 A439 A43B A43D A43F A441	96 78 81 02 26 00 96 79 27 03 BD A2	2 3 3 2 A8	10444	CMPA BNE LDA BEQ JSR	LA449 CINCTR LA444 LA2A8	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE
0586 0587 0588 0589 0590 0591 0592 0593	A435 A437 A439 A43B A43D A43F A441 A444	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF	2 2 3 2 A8	LA444	CMPA BNE LDA BEQ JSR LDB	LA449 CINCTR LA444 LA2A8 #\$FF	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595	A435 A437 A439 A43B A43D A43F A441 A444	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF	2 3 3 3 2 4 8 5 4 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8		CMPA BNE LDA BEQ JSR LDB JSR	LA449 CINCTR LA444 LA2A8 #\$FF LA2AA	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595	A435 A437 A439 A43B A43D A43F A441 A444 A446	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78	2 3 3 3 2 4 8 5 4 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	LA449	CMPA BNE LDA BEQ JSR LDB JSR CLR	LA449 CINCTR LA444 LA2A8 #\$FF	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595 0596	A435 A437 A439 A43B A43D A43F A441 A444	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78	2 3 3 3 2 4 8 5 4 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8		CMPA BNE LDA BEQ JSR LDB JSR	LA449 CINCTR LA444 LA2A8 #\$FF LA2AA	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595 0596 0597	A435 A437 A439 A43B A43D A43F A441 A444 A446	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78	2 3 3 3 2 4 8 5 4 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	LA449 LA44B	CMPA BNE LDA BEQ JSR LDB JSR CLR	LA449 CINCTR LA444 LA2A8 #\$FF LA2AA	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595 0596 0597 0598	A435 A437 A439 A43B A43D A43F A441 A444 A446 A449	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78	2 2 3 3 2 A8 5 2 AA	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS	LA449 CINCTR LA444 LA2A8 #\$FF LA2AA FILSTA	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED
0586 0587 0588 0589 0591 0591 0592 0593 0594 0595 0596 0597 0598 0599	A435 A437 A439 A43B A43D A43F A441 A444 A446 A449 A44B	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78 39	2 A8 2 AA 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	LA449 LA44B	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME
0586 0587 0588 0589 0591 0592 0593 0594 0595 0596 0597 0598	A435 A437 A439 A43B A43D A43F A441 A444 A446 A449 A44B	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78 39 BD A5 9D A5	2 A8 :: 2 AA :: 3	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE
0586 0587 0588 0589 0599 0591 0592 0593 0594 0595 0597 0598 0597 0600 0601 0602	A435 A437 A439 A43B A43D A43F A441 A444 A446 A449 A44B	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78 39 BD A5 9D A5 27 16	2 A8 :: 2 AA :: 3 AA :: 5 78	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR BEQ	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA LA578 GETCCH LA469	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595 0596 0597 0598 0599 0600 0601 0602 0603	A435 A437 A439 A43B A43D A43F A441 A444 A446 A449 A44B	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 0F 78 39 BD A5 9D A5 27 16 BD B2	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR BEQ JSR	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA
0586 0587 0588 0599 0591 0592 0593 0594 0595 0596 0597 0598 0699 0600 0601 06003 0604	A435 A437 A439 A43B A43D A43F A441 A446 A449 A448 A446 A449 A448	96 78 81 02 26 00 96 79 27 03 BD A2 06 FF BD A2 0F 78 39 BD A5 9D A5 27 16 BD B2 C6 41	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR BEQ JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE?
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595 0596 0597 0598 0599 0600 0601 0602 0603	A435 A437 A438 A43B A43B A43F A441 A444 A446 A449 A44B A44B A44C A44F A445 A456 A458	96 78 81 02 26 00 96 79 27 03 BD A2 C6 F7 8D A5 9D A5 27 16 BD A5 27 16 BD B2	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR BEQ JSR	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A
0586 0587 0588 0599 0591 0592 0593 0594 0595 0596 0597 0598 0699 0600 0601 06003 0604	A435 A437 A439 A43B A43D A43F A441 A446 A449 A448 A446 A449 A448	96 78 81 02 26 00 96 79 27 03 BD A2 C6 F7 8D A5 9D A5 27 16 BD A5 27 16 BD B2	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR BEQ JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE?
0586 0587 0588 0589 0590 0591 0592 0593 0594 0595 0596 0597 0598 0690 0601 0602 0603	A435 A437 A438 A43B A43B A43F A441 A444 A446 A449 A44B A44B A44C A44F A445 A456 A458	96 78 81 02 26 00 96 79 27 03 BD A2 06 FF BD A2 07 78 39 BD A5 9D A5 27 16 BD B2 26 EE	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR BEQ JSR LDB JSR	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #*A LB26F	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0699 0600 0601 0602 0603 0604 0606 0606	A435 A437 A439 A43B A43D A43F A441 A444 A446 A449 A44B A44B A448 A453 A456 A458 A458	96 78 81 02 26 00 96 79 27 03 BD A2 0F 78 39 BD A5 9D A5 27 16 BD B2 C6 41 BD B2 26 EE	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR JSR JSR BEQ JSR LDB JSR BNE	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #*A LB26F	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE
0586 0587 0588 0589 0599 0591 0592 0593 0594 0595 0596 0597 0699 0601 0602 0603 0604 0605 0606 0606	A435 A437 A438 A43B A43D A43F A441 A444 A446 A449 A44B A44B A44B A453 A456 A458 A45B A45D	96 78 81 02 26 00 96 79 27 03 BD A2 06 FF BD A2 07 78 39 BD A5 27 16 BD B2 26 41 BD B2 26 EE	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR BEQ JSR LDB JSR BEQ JSR LDB CLR CLR	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0599 0600 0601 0602 0602 0604 0605 0606 0607 0608	A435 A437 A439 A43B A43D A43F A441 A446 A449 A44B A44B A446 A451 A456 A458 A458 A458 A458 A458 A458 A458	96 78 81 02 26 00 96 79 27 03 BD A2 06 FF BD A2 07 78 39 BD A5 27 16 BD B2 26 41 BD B2 26 EE	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR LDB JSR LDB JSR BEQ JSR LDB JSR LDB JSR LDB JSR LDB JSR	LA449 CINCTR LA444 LA2A8 #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0699 0601 0602 0603 0604 0604 0606 0607	A435 A437 A439 A43B A43D A43F A441 A446 A449 A44B A44B A446 A451 A456 A458 A458 A458 A458 A458 A458 A458	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 07 78 39  BD A5 27 16 BD B2 C6 41 BD B2 26 EE 4F BD A6 86 FF 97 6F	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR BEQ JSR LDB JSR LDB JSR LDB JSR BNE CLRA BNE CLRA LDA	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE
0586 0587 0588 0599 0591 0592 0593 0594 0596 0597 0598 0699 0600 0601 0602 0603 0604 0606 0607 0608 0608	A435 A437 A438 A43D A43F A441 A444 A446 A449 A44B A448 A450 A458 A458 A458 A458 A458 A458 A458 A458	96 78 81 02 26 00 96 79 27 03 BD A2 C6 FF BD A2 07 78 39  BD A5 27 16 BD B2 C6 41 BD B2 26 EE 4F BD A6 86 FF 97 6F	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR BEQ JSR LDB JSR BEQ JSR LDB JSR LDB JSR LDB JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0699 0601 0602 0603 0604 0605 0606 0606 0606 0607 0608 0609 0610	A435 A437 A438 A43D A43F A441 A444 A446 A449 A44B A448 A450 A458 A458 A458 A458 A458 A458 A458 A458	96 78 81 02 26 96 79 27 03 8D A2 0F 78 39 8D A5 27 16 8D B2 27 16 8D B2 26 4F 8D A6 86 FF 97 6F 4F	2 A8	LA449 LA44B * CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR BEQ JSR LDB JSR BEQ JSR LDB JSR LDB JSR LDB JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0600 0601 0602 0603 0604 0606 0607 0608 0606 0607 0608 0601 0601 0601 0601 0601 0601 0601	A435 A437 A438 A43D A43F A441 A444 A446 A449 A44B A448 A450 A458 A458 A458 A458 A458 A458 A458 A458	96 78 81 02 26 96 79 27 03 8D A2 0F 78 39 8D A5 27 16 8D B2 27 16 8D B2 26 4F 8D A6 86 FF 97 6F 4F	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR LDB JSR LDB LDB JSR LDB JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0690 0601 0602 0603 0604 0606 0607 0608 0606 0607 0608	A435 A437 A438 A438 A430 A441 A444 A444 A446 A446 A447 A451 A451 A458 A458 A458 A458 A458 A458 A458 A458	96 78 81 02 26 02 60 60 60 60 60 60 60 60 60 60 60 60 60	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR JSR LDB JSR LDA JSR	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE
0586 0587 0588 0599 0591 0592 0593 0594 0596 0597 0598 0599 0600 0601 0602 0603 0604 0605 0606 0607 0608 0609 0611 0612 0613 0611 0612	A435 A437 A438 A438 A430 A441 A444 A444 A448 A445 A451 A453 A458 A458 A458 A463 A466 A466	96 78 81 82 96 77 82 85 86 86 86 86 86 86 86 86 86 86 86 86 86	2 A8 5 78 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ JSR LDB LSR LDA STA LMP LDA CLRA CLRA CLRA CLRA CLRA CLRA CLRA CLR	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM LIST	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE
0586 0587 0588 0589 0590 0591 0592 0593 0596 0596 0597 0598 0600 0601 0602 0603 0604 0604 0606 0607 0608 0607 0608 0601 0610 0611 0612 0613	A435 A437 A438 A430 A441 A444 A446 A448 A448 A448 A450 A458 A458 A458 A458 A458 A466 A468 A466 A468	96 78 81 02 26 80 6 78 81 82 82 88 88 88 88 88 88 88 88 88 88 88	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ LDB BEQ JSR LDB SSR CLR RTS  JSR LDB JSR LDB BEQ JSR LDB JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM LIST /E	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = Ø ZERO OUT ASCII FLAG AND FILE MODE
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0600 0601 0602 0603 0604 0606 0607 0608 0606 0607 0608 0610 0611 0611 0611 0614 0614	A435 A437 A438 A438 A430 A441 A441 A444 A448 A451 A453 A456 A458 A450 A458 A456 A458 A466 A466 A466	96 78 81 82 96 75 82 96 87 88 88 88 86 87 88 88 88 88 88 88 88 88 88 88 88 88	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ JSR LDB BEQ JSR CLR RTS  JSR JSR JSR LDB LDC LCR JSR LDB LCR JSR LDB LCR LDB LCR LDB LDB LCR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM LIST VE ZERO LA65F	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = Ø ZERO OUT ASCII FLAG AND FILE MODE WRITE HEADER BLOCK
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0699 0600 0601 0602 0603 0604 0605 0606 0607 0608 0609 0611 0612 0613 0614 0615 0616 0615	A435 A437 A438 A438 A441 A444 A446 A448 A448 A448 A458 A458 A458 A458 A458	96 78 81 82 27 82 85 86 87 88 88 88 88 88 88 88 88 88 88 88 88	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ JSR LDB JSR CLR RTS  JSR BEQ JSR BEQ JSR BNE CLRA LDA CLRA JMP  CII CSA CLRA LDX CLRA CLRA CLRA CLRA CLRA CLRA CLRA CLRA	LA449 CINCTR LA444 LA2A8 #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB260 #'A LB26F LA44B  LA65C #-1 DEVNUM LIST  ZERO LA65F FILSTA	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = Ø ZERO OUT ASCII FLAG AND FILE MODE WRITE HEADER BLOCK CLOSE FILES
0586 0587 0588 0599 0591 0592 0593 0594 0596 0597 0598 0608 0601 0602 0603 0604 0607 0608 0609 0610 0611 0612 0613 0614 0615 0616	A435 A437 A438 A438 A430 A441 A444 A446 A447 A448 A448 A458 A458 A458 A458 A458 A466 A468 A466 A466 A466 A466 A466 A46	96 78 81 82 82 86 86 87 87 88 88 86 87 87 88 88 88 88 88 88 88 88 88 88 88	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ LDB BEQ JSR LDB JSR CLR RTS  JSR JSR LDB JSR LDB JSR LDB JSR LDB JSR LDB JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM  LIST  ZERO LA65F FILSTA  BLKTYP	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = Ø ZERO OUT ASCII FLAG AND FILE MODE WRITE HEADER BLOCK CLOSE FILES INCREMENT BLOCK NUMBER
0586 0587 0588 0589 0590 0591 0592 0593 0594 0596 0597 0598 0600 0601 0602 0603 0604 0606 0607 0608 0606 0601 0611 0611 0611 0611 0611	A435 A437 A438 A438 A431 A441 A444 A446 A447 A451 A453 A456 A458 A458 A458 A458 A458 A466 A466 A466 A466 A466 A467 A467 A467	96 78 81 82 96 75 81 82 82 84 84 84 84 84 84 84 84 84 84 84 84 84	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ LDA JSR LDB LDB LDB LSR CLR RTS  JSR LDB JSR LDB JSR LDB JSR LDB JSR CLRA JSR	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM  LIST /E  ZERO LA65F FILSTA  BLKTYP WRLDR	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = Ø ZERO OUT ASCII FLAG AND FILE MODE WRITE HEADER BLOCK CLOSE FILES INCREMENT BLOCK NUMBER WRITE 55 S TO CASSETTE
0586 0587 0588 0589 0599 0591 0592 0593 0596 0597 0598 0699 0601 0602 0603 0604 0605 0606 0607 0611 0612 0611 0612 0611 0612 0611 0612 0611 0612 0611 0612	A435 A437 A438 A439 A441 A444 A444 A446 A447 A448 A448 A450 A450 A450 A466 A466 A466 A466 A467 A467 A467 A467	96 78 81 82 27 82 85 86 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 88 88 88 88 88 88 88 88 88 88 88	2 A8	LA449 LA44B * CSAVE CSAVE * NON-ASC LA469	CMPA BNE LDA BDEQ JSR LDB JSR CLR RTS  JSR BEQ JSR BEQ JSR BEQ JSR BNE CLRA LDX	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM LIST //E  ZERO LA65F FILSTA BLKTYP WRLDR TXTTAB	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = Ø ZERO OUT ASCII FLAG AND FILE MODE WRITE HEADER BLOCK CLOSE FILES INCREMENT BLOCK NUMBER WRITE 5 S TO CASSETTE ADDRESS OF PROGRAM START
0586 0587 0588 0599 0591 0592 0593 0594 0596 0597 0598 0699 0601 0602 0603 0604 0604 0607 0608 0609 0611 0612 0613 0614 0615 0616 0617 0618 0618 0619 0619 0619	A435 A437 A438 A438 A441 A444 A444 A444 A448 A445 A451 A453 A456 A456 A456 A466 A466 A466 A466 A467 A467 A467 A473	96 78 81 82 82 88 88 88 88 88 88 88 88 88 88 88	2 A8	LA449 LA44B * CSAVE CSAVE	CMPA BNE LDA BEQ JSR LDB BEQ JSR CLR RTS  JSR BEQ JSR LDB JSR LDB	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM  LIST //E  ZERO LA65F FILSTA BLKTYP WRLDR TXTTAB CBUFAD	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NOME SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = 0 WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = 0 ZERO OUT ASCII FLAG AND FILE MODE WRITE HEADER BLOCK CLOSE FILES INCREMENT BLOCK NUMBER WRITE 55 S TO CASSETTE ADDRESS OF PROGRAM START STORE CURRENT BLOCK START ADDR
0586 0587 0588 0589 0599 0591 0592 0593 0596 0597 0598 0699 0601 0602 0603 0604 0605 0606 0607 0611 0612 0611 0612 0611 0612 0611 0612 0611 0612 0611 0612	A435 A437 A438 A438 A441 A444 A444 A444 A448 A445 A451 A453 A456 A456 A456 A466 A466 A466 A466 A467 A467 A467 A473	96 78 81 82 27 82 85 86 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 86 87 88 88 88 88 88 88 88 88 88 88 88 88	2 A8	LA449 LA44B * CSAVE CSAVE * NON-ASC LA469	CMPA BNE LDA BDEQ JSR LDB JSR CLR RTS  JSR BEQ JSR BEQ JSR BEQ JSR BNE CLRA LDX	LA449 CINCTR LA444 LA2AB #\$FF LA2AA FILSTA  LA578 GETCCH LA469 LB26D #'A LB26F LA44B  LA65C #-1 DEVNUM LIST //E  ZERO LA65F FILSTA BLKTYP WRLDR TXTTAB	NO GET CHARACTER BUFFER CTR WRITE END OF PROG BLOCK IF BUFFER EMPTY WRITE A BLOCK TO TAPE END OF FILE TYPE BLOCK NUMBER WRITE END OF FILE TYPE BLOCK CASSETTE FILE STATUS CLOSED  GO SCAN OFF NAME GET CURRENT CHARACTER IN THE BASIC LINE BRANCH IF NONE SYNTAX ERROR IF NOT COMMA IS THIS AN ASCII SAVE? SYNTAX ERROR IF NOT A RETURN IF NOT END OF LINE FILE TYPE = Ø WRITE OUT HEADER BLOCK CASSETTE CODE SET DEVICE NUMBER TO CASSETTE CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START GO DO A LIST TO CASSETTE  FILE TYPE = Ø ZERO OUT ASCII FLAG AND FILE MODE WRITE HEADER BLOCK CLOSE FILES INCREMENT BLOCK NUMBER WRITE 5 S TO CASSETTE ADDRESS OF PROGRAM START

Ø624	A47C	97 7	n			STA	DIVIEN	DIOCK SIZE
Ø625		DC 1				LDD	BLKLEN VARTAB	BLOCK SIZE END OF PROGRAM
Ø626		93 7					CBUFAD	CURRENT BLOCK STARTING ADDR
Ø627	A482	27 Ø				BEQ	LA491	BRANCH IF IT CAME OUT EXACT
Ø628	A484			FF		CMPD	#255	MORE THAN 255 BYTES LEFT?
Ø629	A488	24 Ø	2			BHS	LA48C	YES
0630	A48A	D7 7	D			STB	BLKLEN	USE ACTUAL BLOCK SIZE IF LESS THAN 255
0631		BD A			LA48C	JSR	SNDBLK	WRITE BLOCK TO CASSETTE
Ø632		2Ø E				BRA	LA478	DO ANOTHER BLOCK
Ø633		00 7			LA491	NEG	BLKTYP	MAKE BLOCK NUMBER NEGATIVE (EOF BLOCK)
Ø634		ØF 7				CLR	BLKLEN	ZERO BLOCK SIZE
Ø635 Ø636	A495	7E A	/ E/			JMP	LA7E7	WRITE A BLOCK, TURN OFF MOTOR
Ø637					* CLOAD			
	A498	ØF 78	В		CLOAD	CLR	FILSTA	CLOSE FILES
	A49A					CMPA		IS IT CLOADM?
0640	A49C	27 6	Ø			BEQ	LA4FE	BRANCH IF SO
0641	A49E	32 6	2			LEAS	2,S	GET RID OF THE RETURN
0642	A4AØ	BD A	5 C5			JSR	LA5C5	GO GET FILE NAME
	A4A3					JSR	LA648	SEARCH FOR FILE
	A4A6					TST	CASBUF+1Ø	GET FILE MODE (NON-ZERO=DATA OR ASCII)
	A4A9					BEQ	LA4C8	ZERO = CRUNCHED BASIC OR MACHINE LANG
Ø646 Ø647	A4AB	27 1				LDA BEQ	CASBUF+9 LA4CD	GET ASCII FLAG  BAD FILE NODE Ø = CRUNCHED OR MACH LANG
	A4BØ					JSR	LAD19	DO A NEW
Ø649	A4B3					LDA	#-1	TAPE DEVICE NUMBER
	A4B5					STA	DEVNUM	SET DEVICE NUMBER TO TAPE
Ø651	A4B7					INC	FILSTA	FILE TYPE = INPUT
Ø652	A4B9	BD A	6 35			JSR	LA635	GO LOAD ASCII RECORD
Ø653	A4BC	7E A	C 7C			JMP	LAC7C	GO LOAD AND CRUNCH INPUT
Ø654								
Ø655							1 BASIC S DIRECT LOOP IF CO	NSOLE
Ø656					* IN BUFF			
Ø657	A4BF	BD Ø			LA4BF	JSR	RVEC13	HOOK INTO RAM
Ø658		BD A				JSR	LA42D	CLOSE ACTIVE FILE
Ø659 Ø66Ø	A4U5	7E A	L /3			JMP	LAC73	GO TO BASIC S DIRECT LOOP
Ø661					* CLOAD A	CDIINCE	IED BASIC	
Ø662	A4C8	B6 Ø	1 F2		LA4C8	LDA	CASBUF+8	FILE TYPE
Ø663	A4CB				271.00	BEQ	LA4DØ	ZERO IS CSAVE TYPE
Ø664	A4CD				LA4CD	JMP	LA616	BAD FILE MODE IF NOT BASIC FILE
Ø665	A4DØ				LA4DØ	JSR	LAD19	DO A NEW
Ø666	A4D3	BD A	7 7C			JSR	CASON	TURN ON TAPE, START READING
Ø667	A4D6	9E 1	9			LDX	TXTTAB	GET START OF PROGRAM ADDRESS
Ø668	A4D8				LA4D8	STX	CBUFAD	STORE IT IN LOAD BUFFER
Ø669		DC 7	E			LDD	CBUFAD	GET START ADDRESS TO D REG
Ø67Ø		4C				INCA		ADD 256 TO LOAD ADDRESS
Ø671		BD A				JSR	LAC37	SEE IF ROOM BELOW STACK FOR ONE BLOCK
Ø672 Ø673	A4E3	BD A				JSR BNE	GETBLK LA4F8	READ A BLOCK
Ø674	A4E5					LDA	BLKTYP	GOT AN ERROR DURING READ BLOCK NUMBER
Ø675	A4E7					BEQ	LA4F8	I/O ERROR IF HEADER BLOCK TYPE
Ø676	A4E9					BPL	LA4D8	REAR MORE IF BLOCK NUMBER POSITIVE
Ø677	A4EB	9F 1	В			STX	VARTAB	SET END OF PROGRAM ADDRESS
Ø678	A4ED	8D 4	C			BSR	LA53B	TURN OFF TAPE DECK
Ø679	A4EF	8E A	B EC			LDX	#LABED-1	POINT TO OK MESSAGE
Ø68Ø		BD B				JSR	LB99C	PRINT OK TO CONSOLE OUT
0681	A4F5	7E A	C E9			JMP	LACE9	RESET INPUT POINTER, CLEAR VARIABLES AND
Ø682					*			RETURN TO MAIN LOOP OF BASIC
Ø683	4.450	DD 41			1.4450	100	14010	DO A NEW
Ø684		BD A				JSR		DO A NEW
Ø685 Ø686	A4FB	/E A	0 19		LA4FB	JMP	LA619	I/O ERROR
Ø687					* CLOADM			
Ø688	A4FE	9D 9	F		LA4FE	JSR	GETNCH	GET NEXT CHARACTER IN BASIC LINE
Ø689							LA578	GO SCAN OFF NAME
Ø69Ø							LA648	SEARCH FOR FILE
0691	A5Ø5	9E 8	A			LDX	ZERO	STORE ZERO TO X REG, DEFAULT OFFSET VALUE
Ø692						JSR	GETCCH	CHECK FOR AN OFFSET
Ø693							LA511	BRANCH IF NO OFFSET
Ø694							LB26D	SYNTAX CHECK FOR COMMA EVALUATE OFFSET; RETURN VALUE IN X
Ø695							LB73D	
					LA511	IΠΔ		CHECK FILE MODE
aco-	A511							
	A514	81 Ø	2			CMPA	#2	IS IT MACHINE LANGUAGE?
Ø698	A514 A516	81 Ø: 26 B:	2 5			CMPA BNE	#2 LA4CD	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT
Ø698 Ø699	A514 A516 A518	81 Ø: 26 B: FC Ø:	2 5 1 E5			CMPA BNE LDD	#2 LA4CD CASBUF+11	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE
0698 0699 0700	A514 A516 A518 A51B	81 Ø: 26 B: FC Ø: 33 8:	2 5 1 E5 B			CMPA BNE LDD LEAU	#2 LA4CD CASBUF+11 D,X	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET
0698 0699 0700 0701	A514 A516 A518 A51B A51D	81 Ø: 26 B: FC Ø: 33 8! DF 9!	2 5 1 E5 B			CMPA BNE LDD LEAU STU	#2 LA4CD CASBUF+11 D,X EXECJP	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT
0698 0699 0700	A514 A516 A518 A51B A51D A51F	81 Ø3 26 B3 FC Ø3 33 81 DF 91 7D Ø3	2 5 1 E5 B D 1 E4			CMPA BNE LDD LEAU STU TST	#2 LA4CD CASBUF+11 D,X EXECJP	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET
0698 0699 0700 0701 0702	A514 A516 A518 A51B A51D A51F A522	81 Ø: 26 B: FC Ø: 33 8! DF 9! 7D Ø: 26 A:	2 5 1 E5 B D 1 E4			CMPA BNE LDD LEAU STU TST	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE
0698 0699 0700 0701 0702 0703	A514 A516 A518 A51B A51D A51F A522 A524	81 Ø3 26 B3 FC Ø3 33 81 DF 91 7D Ø3 26 A3 FC Ø3	2 5 1 E5 B D 1 E4 9			CMPA BNE LDD LEAU STU TST BNE LDD	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD CASBUF+13	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE BAD FILE MODE ERROR
0698 0699 0700 0701 0702 0703 0704 0705 0706	A514 A516 A518 A51B A51D A51F A522 A524 A527 A529	81 Ø 81 81 81 82 82 82 82 82 82 82 82 82 82 82 82 82	2 5 1 E5 B D 1 E4 9 1 E7 B			CMPA BNE LDD LEAU STU TST BNE LDD	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD CASBUF+13 D,X	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE BAD FILE MODE ERROR GET LOAD ADDR FROM TAPE
0698 0699 0700 0701 0702 0703 0704 0705 0706	A514 A516 A518 A51B A51D A51F A522 A524 A527 A529 A52B	81 Ø. 26 B. FC Ø. 33 8. DF 9. 7D Ø. 26 A. FC Ø. 3Ø 8. 9F 7. BD A.	2 5 1 E5 B D E4 9 1 E7 B E 7 7C			CMPA BNE LDD LEAU STU TST BNE LDD LEAX STX JSR	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD CASBUF+13 D,X CBUFAD CASON	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE BAD FILE MODE ERROR GET LOAD ADDR FROM TAPE ADD OFFSET STORE IN BUFFER START ADDRESS POINTER START UP TAPE
0698 0699 0700 0701 0702 0703 0704 0705 0706 0707 0708	A514 A516 A518 A51B A51D A51F A522 A524 A527 A529 A52B A52E	81 Ø. 26 B. FC Ø. 33 8. DF 9. 7D Ø. 26 A. FC Ø. 3Ø 8. 9F 7. BD A. BD A.	2 5 1 E5 B D E4 9 1 E7 B E 7 7C 7 ØB			CMPA BNE LDD LEAU TST BNE LDD LEAX STX JSR JSR	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD CASBUF+13 D,X CBUFAD CASON GETBLK	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE BAD FILE MODE ERROR GET LOAD ADDR FROM TAPE ADD OFFSET STORE IN BUFFER START ADDRESS POINTER START UP TAPE READ A BLOCK
0698 0699 0700 0701 0702 0703 0704 0705 0706 0707 0708 0709	A514 A516 A518 A51B A51D A51F A522 A524 A527 A529 A52B A52E A531	81 Ø1 26 B1 FC Ø1 33 81 DF 91 7D Ø1 26 A2 FC Ø3Ø 81 9F 71 BD A1 BD A2 26 C1	2 5 1 E5 B D 1 E4 9 1 E7 B E 7 7C 7 ØB			CMPA BNE LDD LEAU STU TST BNE LDD LEAX STX JSR JSR BNE	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD CASBUF+13 D,X CBUFAD CASON GETBLK LA4FB	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE BAD FILE MODE ERROR GET LOAD ADDR FROM TAPE ADD OFFSET STORE IN BUFFER START ADDRESS POINTER START UP TAPE READ A BLOCK BRANCH IF I/O ERROR
0698 0699 0700 0701 0702 0703 0704 0705 0706 0707 0708 0709	A514 A516 A518 A51B A51D A51F A522 A524 A527 A529 A528 A52E A531 A533	81 Ø1 26 B1 FC Ø 33 81 DF 91 7D Ø2 26 A2 FC Ø 3Ø 81 9F 71 BD A BD A 26 CG 9F 7	2 5 1 E5 B D 1 E4 9 1 E7 B E 7 7C 7 ØB 8			CMPA BNE LDD LEAU STU TST BNE LDD LEAX STX JSR JSR BNE STX	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD CASBUF+13 D,X CBUFAD CASON GETBLK LA4FB CBUFAD	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE BAD FILE MODE ERROR GET LOAD ADDR FROM TAPE ADD OFFSET STORE IN BUFFER START ADDRESS POINTER START UP TAPE READ A BLOCK BRANCH IF I/O ERROR STORE NEW START ADDR (ONE BLOCK HIGHER)
0698 0699 0700 0701 0702 0703 0704 0705 0706 0707 0708 0709	A514 A516 A518 A51B A51D A51F A522 A524 A527 A529 A528 A52E A531 A533	81 Ø: 26 B: FC Ø 33 8i DF 9i 7D Ø 26 A: FC Ø 3Ø 8i 9F 7i BD A: BD A: 26 Ci 9F 7i ØD 7i	2 5 1 E5 B D E4 9 1 E7 B E 7 7C 7 ØB B E C			CMPA BNE LDD LEAU STU TST BNE LDD LEAX STX JSR JSR BNE	#2 LA4CD CASBUF+11 D,X EXECJP CASBUF+10 LA4CD CASBUF+13 D,X CBUFAD CASON GETBLK LA4FB CBUFAD BLKTYP	IS IT MACHINE LANGUAGE? BAD FILE MODE ERROR IF NOT GET TRANSFER ADDR FROM TAPE ADD OFFSET STORE TRANSFER ADDR IN EXEC ARGUMENT CHECK FILE MODE BAD FILE MODE ERROR GET LOAD ADDR FROM TAPE ADD OFFSET STORE IN BUFFER START ADDRESS POINTER START UP TAPE READ A BLOCK BRANCH IF I/O ERROR

Ø713	A539	2A	F3				BPL	LA52E	GO READ SOME MORE
0714	A53B	7E .	Α7	E9		LA53B	JMP	LA7E9	GO TURN OFF TAPE DECK
Ø715						+ 5750			
Ø716 Ø717	A53E	27	Ø5			* EXEC EXEC	BEQ	LA545	BRANCH IF NO ARGUMENT
Ø718	A54Ø	BD		3 D			JSR	LB73D	EVALUATE ARGUMENT - ARGUMENT RETURNED IN X
0719	A543	9F					STX	EXECJP	STORE X TO EXEC JUMP ADDRESS
0720 0721	A545	6E	9F	ØØ	9 D	LA545	JMP	[EXECJP]	GO DO IT
Ø721						* BREAK C	HECK		
Ø723	A549	BD	Ø1	7 F		LA549	JSR	RVEC11	HOOK INTO RAM
0724	A54C	96	6F				LDA	DEVNUM	GET DEVICE NUMBER
Ø725 Ø726	A54E A54F	4C 27	Eα				INCA BEQ	LA5A1	CHECK FOR TAPE RETURN IF TAPE
Ø727	A551	7E .		ЕВ			JMP	LADEB	GO DO BREAK KEY CHECK
Ø728									
Ø729								EVALUATES AN ARGUMENT	TOTO DICELAY DAM
0730 0731	A554	BD	В3	F4		LASS4	JSR	E IT IS WITHIN LIMITS OF V LB3E4	EVALUATE EXPRESSION AND RETURN VALUE IN ACCD
0732	A557	83	Ø1	FF			SUBD		ONLY 512 VIDEO DISPLAY LOCATIONS
Ø733		10			EC			LB44A	BRANCH IF > 511 TO ILLEGAL FUNCTION CALL
Ø734 Ø735	A55E A561	C3		FF			ADDD STD	#VIDRAM+511 CURPOS	ADD BACK IN OFFSET + START OF VIDEO RAM PUT THE CURSOR THERE
Ø736	A563	39	00				RTS	CURPUS	FOI THE CORSON THERE
Ø737									
Ø738						* INKEY\$		*******	
Ø739 Ø74Ø	A564 A566	96 26				INKEY	LDA BNE	IKEYIM LA56B	WAS A KEY DOWN IN THE BREAK CHECK? YES
0741	A568	BD .		СВ			JSR	KEYIN	GO GET A KEY
0742	A56B	ØF				LA56B	CLR	IKEYIM	CLEAR INKEY RAM IMAGE
Ø743	A56D	97			1.0		STA	FPAØ+3	STORE THE KEY IN FPAØ
Ø744 Ø745	A56F A573	1Ø 97		11	10		STA	LB68F STRDES	CONVERT FPAØ+3 TO A STRING SET LENGTH OF STRING = Ø IF NO KEY DOWN
Ø746	A575	7 E		9B			JMP	LB69B	PUT A NULL STRING ONTO THE STRING STACK
0747									
Ø748 Ø749	A E 7 O	0.5	α1	D 1		* STRIP A LA578	FILEN.	AME OFF OF THE BASIC INPUT #CFNBUF	LINE POINT TO FILE NAME BUFFER
Ø749 Ø75Ø	A578 A57B	8E 6F		υı		LAS/6	CLR	,X+	CLEAR THE FIRST BYTE - IT WILL CONTAIN THE COUNT
0751						*		,	OF THE NUMBER OF CHARACTERS IN THE NAME
Ø752	A57D	86					LDA	#SPACE	SPACE
Ø753 Ø754	A57F A581	A7 8C		DΛ		LA57F	STA CMPX	,X+ #CASBUF	BLANK FILL 8 CHARS DONE?
Ø755	A584	26		υ <b>Λ</b>			BNE	LA57F	NO
Ø756	A586	9D .	Α5				JSR	GETCCH	GET CURRENT INPUT CHAR
Ø757	A588	27					BEQ	LA5A1	RETURN IF NO NAME
Ø758 Ø759	A58A A58D	BD BD					JSR JSR	LB156 LB654	GET THE FILE NAME - EVALUATE EXPRESSION POINT X TO START OF NAME (TOP STRING ON STRING STACK)
Ø76Ø	A59Ø	CE					LDU	#CFNBUF	CASSETTE FILE NAME BUFFER
0761	A593	E7	CØ				STB	,U+	STORE THE NUMBER OF BYTES IN THE NAME
Ø762	A595	27	ØA				BEQ	LA5A1	NULL NAME (BLANK NAME)
Ø763 Ø764	A597 A597	8C C6	as			LA598	FCB LDB	SKP2 #8	SKIP THE NEXT TWO BYTES MOVE 8 BYTES
Ø765	N337		50			ENSSO	LUU	#0	11072 0 01723
Ø766								ES FROM (X) TO (U)	
Ø767 Ø768	A59A A59C	A6 A7				LA59A	LDA STA	, X+ , U+	GET BYTE FROM X STORE IT AT U
Ø769	A59E	5A	CB				DECB	,0+	MOVED ALL BYTES?
0770	A59F	26	F9				BNE	LA59A	NO
0771	A5A1	39				LA5A1	RTS		
Ø772 Ø773						* GET DEV	TCE NIII	MBER FROM BASIC LINE - CHE	CK AVITUITA
Ø774	A5A2	BD	В2	6 D		LA5A2	JSR	LB26D	CHECK FOR COMMA, SYNTAX ERROR IF NONE
Ø775	A5A5	81	23			LA5A5	CMPA	#'#	IS NEXT CHARACTER A NUMBER?
Ø776		26					BNE	LA5AB	NO CET NEVI PASIC INDUT CHADACTED
Ø777 Ø778	A5A9 A5AB			41		LA5AB	JSR JSR	GETNCH LB141	GET NEXT BASIC INPUT CHARACTER EVALUATE EXPRESSION
Ø779	A5AE					LASAE	JSR	INTCNV	CONVERT FPAØ TO INTEGER, RETURN VALUE IN ACCD
0780	A5B1						ROLB		MSB OF ACCB TO CARRY
Ø781	A5B2						ADCA		ADD MSB OF ACCB TO ACCA
Ø782 Ø783	A5B4 A5B6		09				BNE RORB	LA61F	DEVICE # ERROR IF ACCA <ff80 or="">007F RESTORE ACCB</ff80>
Ø784	A5B7		6 F				STB	DEVNUM	STORE B IN DEVICE NUMBER
Ø785	A5B9			61			JSR	RVEC1	HOOK INTO RAM
Ø786 Ø787	A5BC A5BE						BEQ BPL	LA5C4 LA61F	BRANCH IF DEVICE NUMBER SET TO SCREEN DEVICE NUMBER ERROR IF POSITIVE DEVICE NUMBER
Ø788	A5CØ							#-2	LOWEST LEGAL DEVICE NUMBER
Ø789	A5C2						BLT	LA61F	DEVICE NUMBER ERROR
0790	A5C4	39				LA5C4	RTS		
Ø791 Ø792						** TUTC D	OUTINE	WILL SCAN OFF THE FILE NA	ME EDOM A DASIC LINE
Ø792 Ø793								SYNTAX ERROR IF THERE ARE	
Ø794								E END OF THE NAME	
Ø795	A5C5					LA5C5	BSR	LA578	SCAN OFF NAME
Ø796 Ø797	A5C7 A5C9					LA5C9	JSR BEQ	GETCCH LA5C4	GET CURRENT INPUT CHAR FROM BASIC LINE RETURN IF END OF LINE
Ø798	A5CB			77		LHJ03	JMP	LB277	SYNTAX ERROR IF ANY MORE CHARACTERS
Ø799									
Ø8ØØ	AFCE	D.D.	a ı	00		* E0F	100	DVEC14	HOOV INTO DAM
0801	A5CE	DU	υI	00		EOF	JSR	RVEC14	HOOK INTO RAM

Ø8Ø2	A5D1	96 6	2 5			LDA	DEVNUM	GET DEVICE NUMBER
Ø8Ø3	A5D1					PSHS	A	
Ø8Ø4	A5D5					BSR	LA5AE	SAVE IT CHECK DEVICE NUMBER
Ø8Ø5	A5D5	BD A		ED		JSR	LASED	CHECK FOR PROPER FILE AND MODE
Ø8Ø6		5F	٦,5	LU		CLRB	LASED	NOT EOF FLAG = Ø
Ø8Ø7	A5DB		6 F			LDA	DEVNUM	TEST DEVICE NUMBER
Ø8Ø8	A5DD					BEQ	LA5E4	BRANCH IF NOT SET TO DISPLAY
Ø8Ø9	A5DF					TST	CINCTR	ANY CHARACTERS LEFT TO SEND?
Ø81Ø		26 8				BNE	LA5E4	YES
Ø811	A5E3		01			COMB	ENJET	NO - EOF: SET FLAG = -1 (\$FF)
Ø812	A5E4		712		LA5E4	PULS	A	GET DEVICE NUMBER BACK AGAIN
Ø813	A5E6				2,1021	STA	DEVNUM	RESTORE IT
Ø814	A5E8				LA5E8	SEX	DETROIT	CONVERT ACCB TO 2 DIGIT SIGNED INTEGER
Ø815	A5E9		R4	F4	2,1020	JMP	GIVABF	CONVERT ACCD TO FLOATING POINT
Ø816								
Ø817					* SKIPF			
Ø818	A5EC	8D [	D7		SKIPF	BSR	LA5C5	SCAN OFF THE BASIC FILE NAME
Ø819	A5EE					BSR	LA648	LOOK FOR THAT FILE ON TAPE
Ø82Ø	A5FØ			D1		JSR	LA6D1	READ THE FILE
Ø821	A5F3					BNE	LA619	I/O ERROR
Ø822	A5F5	39				RTS		
Ø823								
Ø824					* OPEN			
Ø825	A5F6	BD @	ð1	5 E	OPEN	JSR	RVECØ	HOOK INTO RAM
Ø826	A5F9	BD E	В1	56		JSR	LB156	GET FILE STATUS (INPUT,OUTPUT)
Ø827	A5FC	BD E	B6	A4		JSR	LB6A4	GET FIRST BYTE OF STATUS STRING TO ACCB
Ø828	A5FF	34 8	<b>0</b> 4			PSHS	В	SAVE IT ON STACK
Ø829	A6Ø1	8D 9	9 F			BSR	LA5A2	CHECK FOR SYNTAX AND GET DEVICE NUMBER
0830	A6Ø3	BD E	B2	6D		JSR	LB26D	SYNTAX CHECK FOR COMMA, SYNTAX ERROR IF NOT
Ø831	A6Ø6					BSR	LA5C5	GET FILE NAME
Ø832	A6Ø8	96 6	6F			LDA	DEVNUM	GET DEVICE NUMBER
Ø833	A6ØA	ØF 6	6F			CLR	DEVNUM	SET DEVICE NUMBER TO SCREEN
Ø834	A6ØC	35 @	<b>0</b> 4			PULS	В	GET STATUS AGAIN
Ø835	A6ØE	C1 4	49			CMPB	#'I	IS IT INPUT MODE?
Ø836	A61Ø					BEQ	LA624	YES
Ø837	A612	C1 4	4 F			CMPB	#'0	IS IT OUTPUT MODE?
Ø838	A614	27 4	42			BEQ	LA658	YES
Ø839								
Ø84Ø							IPUT OR OUTPUT, BAD FILE MO	
0841	A616	C6 2	2 A		LA616	LDB	#21*2	ERROR # 21 BAD FILE MODE
Ø842	A618					FCB	SKP2	SKIP TWO BYTES
Ø843	A619		28		LA619	LDB	#20*2	ERROR # 20 I/O ERROR
Ø844	A61B					FCB	SPK2	SKIP TWO BYTES
Ø845	A61C		24		LA61C	LDB	#18*2	ERROR # 18 FILE ALREADY OPEN
Ø846	A61E					FCB	SKP2	SKIP TWO BYTES
Ø847	A61F				LA61F	LDB	#19*2	ERROR # 19 DEVICE NUMBER ERROR
Ø848	A621	7E /	AC	46		JMP	LAC46	JUMP TO ERROR HANDLER
Ø849	1624	4.0			*	TNCA		DEVICE NUMBER OFF TO TARES
Ø85Ø	A624	4C			LA624	INCA	1 4 6 1 6	DEVICE NUMBER SET TO TAPE?
Ø851 Ø852	A625 A627					BMI	LA616 LA657	BAD FILE MODE IF DEVNUM = NEG BUT NOT CASSETTE RETURN IF DEVNUM WAS SET TO SCREEN OR DISK
Ø853	A027	20 2	2.0		* SET TO	BNE	LA057	RETURN IF DEVNOM WAS SET TO SCREEN OR DISK
Ø854	A629	ΩD 1	1 D		" 3E1 10	BSR	LA648	GET HEADER BLOCK
Ø855		B6 6		E3		LDA	CASBUF+9	GET ASCII FLAG
Ø856		B4 @				ANDA	CASBUF+1Ø	AND IT WITH FILE MODE
						,,,,,		
			F3			BFO	LAblb	BAD FILE MODE - CRUNCHED FILE OR MACH LANG
Ø857	A631	27 E				BEQ INC	LA616 FILSTA	BAD FILE MODE - CRUNCHED FILE OR MACH LANG OPEN FILE FOR INPUT
Ø857 Ø858	A631 A633	27 E ØC 7	78	Ø1	LA635	INC	FILSTA	OPEN FILE FOR INPUT
Ø857 Ø858 Ø859	A631 A633 A635	27 E ØC 7 BD A	78 47	Ø1	LA635	INC JSR	FILSTA LA701	OPEN FILE FOR INPUT START TAPE, READ A BLOCK
Ø857 Ø858	A631 A633 A635	27 E ØC 7	78 47 DF	Ø1	LA635	INC	FILSTA	OPEN FILE FOR INPUT
Ø857 Ø858 Ø859 Ø86Ø	A631 A633 A635 A638	27 E ØC 7 BD 4 26 E ØD 7	78 A7 DF 7C	Ø1	LA635	INC JSR BNE TST	FILSTA LA7Ø1 LA619	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR
Ø857 Ø858 Ø859 Ø86Ø Ø861	A631 A633 A635 A638 A63A	27 E ØC 7 BD 4 26 E ØD 7 27 E	78 A7 DF 7C DB	01	LA635	INC JSR BNE TST	FILSTA LA7Ø1 LA619 BLKTYP	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER
Ø857 Ø858 Ø859 Ø86Ø Ø861 Ø862	A631 A633 A635 A638 A63A A63C	27 E ØC 7 BD A 26 E ØD 7 27 E 28 1	78 A7 DF 7C DB	Ø1	LA635	INC JSR BNE TST BEQ	FILSTA LA7Ø1 LA619 BLKTYP LA619	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK
Ø857 Ø858 Ø859 Ø86Ø Ø861 Ø862 Ø863	A631 A633 A635 A638 A63A A63C A63E	27 E ØC 7 BD A 26 E ØD 7 27 E 28 1	78 A7 DF 7C DB 17	Ø1	LA635	INC JSR BNE TST BEQ BMI	FILSTA LA701 LA619 BLKTYP LA619 LA657	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK
0857 0858 0859 0860 0861 0862 0863 0864	A631 A633 A635 A638 A63A A63C A63E A64Ø A642	27 E ØC 7	78 A7 DF 7C DB 17 7D F1	Ø1	LA635 LA644	INC JSR BNE TST BEQ BMI LDA	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT
0857 0858 0859 0860 0861 0862 0863 0864	A631 A633 A635 A638 A63A A63C A63E A64Ø A642	27 E ØC 7	78 A7 DF 7C DB 17 7D F1	01		INC JSR BNE TST BEQ BMI LDA BEQ	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK
Ø857 Ø858 Ø859 Ø860 Ø861 Ø862 Ø863 Ø864 Ø865	A631 A633 A635 A638 A63A A63C A63E A64Ø A642	27 E ØC 7	78 A7 DF 7C DB 17 7D F1	01		INC JSR BNE TST BEQ BMI LDA BEQ STA	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR I/O ERROR I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866	A631 A633 A635 A638 A63A A63C A63E A64Ø A642	27 E ØC 7	78 A7 DF 7C DB 17 7D F1	01	LA644	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR I/O ERROR I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867	A631 A633 A635 A638 A63A A63C A63E A640 A642 A644 A646	27 E ØC 7 BD 4 26 E ØD 7 27 E 28 1 96 7 27 F 97 7 20 Ø	78 A7 DF 7C DB 117 7D F1 79 ØA	01	LA644	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR LA652	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR I/O ERROR I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0868	A631 A633 A635 A638 A63A A63C A63E A64Ø A642 A644 A646	27 E	78 A7 DF 7C DB 117 7D F1 79 ØA	01	LA644 * SEARCH	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA61C	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0868	A631 A633 A635 A638 A63A A63C A63E A640 A642 A644 A646	27 E	78 A7 DF 7C DB 117 7D F1 79 ØA	01	LA644 * SEARCH	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN?
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0868	A631 A633 A635 A638 A63A A63C A63E A640 A642 A644 A646	27 E ØC 7 F BD # A 26 E ØC 7 F 97 7 7 20 ØC 8 B 3 26 ØC 8 C 8 B 3 26 ØC 7 F B	78 A7 DF 7C DB 117 7D F1 77 ØA	01	LA644 * SEARCH	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA61C LA6611 LA6619	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME I/O ERROR
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0868 0870 0871 0871	A631 A633 A635 A638 A63A A63C A648 A644 A646 A648 A648 A648 A646 A648	27 E ØC 7 F ØC 7 F ØC 9 F ØC 9 F 7 F ØC 9 F F ØC 9 F ØC 9 F F F ØC 9 F F F ØC 9 F F F F ØC 9 F F F F F F F F F F F F F F F F F F	78 A7 DF 7C DB 117 7D F1 77 D Ø A		LA644  * SEARCH LA648	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FII TST BNE BSR BNE CLR	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER
0857 0858 0859 0860 0861 0862 0863 0866 0867 0866 0867 0868 0871 0871 0872 0873	A631 A633 A635 A638 A638 A63C A63E A64Ø A642 A644 A646 A648 A64A A64C A64E A65Ø A652	27 E 0C 7	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644 * SEARCH LA648	INC JSR BNE TST BEQ STA BRA FOR FIL TST BNE BSR BSR BNE CLR LDX	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA610 LA610 CINCTR CA681 LA619 CINCTR #CASBUF	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME I/O ERROR
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0878 0878 0879 0871 0872 0873	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE BSR BNE CLR LDX STX	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0878 0878 0879 0871 0872 0873 0874 0875	A631 A633 A635 A638 A638 A63C A63E A64Ø A642 A644 A646 A648 A64A A64C A64E A65Ø A652	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648	INC JSR BNE TST BEQ STA BRA FOR FIL TST BNE BSR BSR BNE CLR LDX	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA610 LA610 CINCTR CA681 LA619 CINCTR #CASBUF	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0868 0867 0871 0872 0871 0874 0875	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE BSR BNE CLR LDX STX RTS	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS
0857 0858 0859 0860 0861 0862 0863 0864 0866 0867 0869 0870 0871 0872 0873 0874 0876 0876	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE BSR BNE CLR LDX STX RTS	FILSTA LA701 LA619 BLKTYP LA619 LA657 BLKLEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA610 LA610 CINCTR CA681 LA619 CINCTR #CASBUF	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS
0857 0858 0859 0860 0861 0862 0863 0864 0866 0867 0887 0871 0872 0873 0874 0875 0877	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE BSR CLR LDX STX RTS	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME I/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT
0857 0858 0859 0860 0861 0862 0863 0864 0865 0867 0868 0867 0871 0871 0877 0878 0877 0878 0877	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE BSR BSR BNE CLR LDX RTS UT THE	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0868 0867 0871 0872 0873 0874 0875 0876 0877	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FIL TST BNE BSR BNE CLR LDX STX RTS UT THE	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE NAME FILE TYPE
0857 0858 0859 0860 0861 0862 0863 0864 0866 0867 0868 0870 0871 0873 0874 0875 0877 0876 0877	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA65Ø LA652 LA657  * WRITE 0  ** CASBUF ** CASBUF ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FII TST BNE BNE CLR LDX STX RTS UT THE	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE TYPE ASCII FLAG
0857 0858 0859 0860 0861 0862 0863 0864 0866 0867 0868 0871 0872 0873 0874 0875 0877 0878 0877 0878 0878 0871	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF ** CASBUF ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ BMI LDA BERA STA BNE BRA BNE CLR LDX STX STX STX STX STX STX STX STX STX ST	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE TYPE ASCII FLAG FILE MODE
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0871 0872 0871 0872 0873 0874 0876 0877 0878 0879 08879 08879 08881 08881 08883 08883	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF ** CASBUF ** CASBUF ** CASBUF ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FII TST BNE BSR BNE LDX STX RTS UT THE +8 +9 +111,12	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE TYPE ASCII FLAG FILE MODE TRANSFER ADDRESS
0857 0858 0859 0860 0861 0862 0863 0864 0866 0867 08870 0871 0872 0873 0874 0875 0876 0877 0878 0879 0879 0889 0889 0889 0888	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF ** CASBUF ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FII TST BNE BSR BNE LDX STX RTS UT THE +8 +9 +111,12	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE TYPE ASCII FLAG FILE MODE
0857 0858 0859 0860 0861 0862 0863 0864 0866 0867 0871 0871 0872 0873 0874 0877 0877 0878 0878 0878 0881 0888 0888	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 7 C 7 C 8 D 8 D 8 D 8 C 8 D 9 F 7 C 8 D 8 D 8 D 9 F 7 C 8 D 9 F 7 C 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8 D 8	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF ** CASBUF ** CASBUF ** CASBUF ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FILL BNE BSR BSR CLR LDX TT THE LDX STX STX STX STX STX STX STX STX STX ST	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA61C LA661 LA61C LA681 LA619 CINCTR #CASBUF CINPTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME I/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE TYPE ASCII FLAG FILE MODE TRANSFER ADDRESS START ADDRESS
0857 0858 0859 0860 0861 0862 0863 0864 0865 0866 0867 0871 0871 0872 0873 0874 0877 0878 0878 0879 0881 0882 0883 0884 0885	A631 A633 A638 A638 A63A A63C A649 A642 A644 A644 A646 A645 A657	27 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6 C 7 E 6	78 A7 DF 7C DB 117 7D F1 79 ØA 78 DØ 333 C9 79		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF  ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ BMI LDA BEQ STA BRA FOR FIII TST BNE BSR CLR LDX STX RTS UT THE	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA662 E NAME IN CNMBUF FILSTA LA61C LA661 LA619 CINCTR #CASBUF CINCTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK 1/O ERROR CHECK BLOCK NUMBER 1/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME 1/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE TYPE ASCII FLAG FILE TYPE ASCII FLAG FILE MODE TRANSFER ADDRESS START ADDRESS
0857 0858 0859 0860 0861 0862 0863 0864 0866 0867 0871 0871 0872 0873 0874 0877 0877 0878 0878 0878 0881 0888 0888	A631 A633 A635 A638 A63A A63C A63E A644 A644 A646 A644 A646	27 E 6 6 7 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7	78 77 77 77 78 78 78 78 78 78 78 77 78		LA644  * SEARCH LA648  LA650 LA652 LA657  * WRITE 0  ** CASBUF ** CASBUF ** CASBUF ** CASBUF ** CASBUF	INC JSR BNE TST BEQ BMI LDA BEQ STA BRA FOR FILL BNE BSR BSR CLR LDX TT THE LDX STX STX STX STX STX STX STX STX STX ST	FILSTA LA701 LA619 BLKTYP LA619 LA657 BIKKEN LA635 CINCTR LA652 E NAME IN CNMBUF FILSTA LA61C LA661 LA61C LA681 LA619 CINCTR #CASBUF CINPTR	OPEN FILE FOR INPUT START TAPE, READ A BLOCK I/O ERROR CHECK BLOCK NUMBER I/O ERROR IF HEADER BLOCK BRANCH IF THIS IS THE LAST BLOCK CHAR COUNT READ ANOTHER BLOCK IF NULL BLOCK STORE IN TEMP CHARACTER COUNTER RESET BUFFER POINTER  IS THE FILE OPEN? YES- FILE ALREADY OPEN SEARCH FOR CORRECT FILE NAME I/O ERROR CLEAR CHARACTER COUNTER CASSETTE INPUT BUFFER ADDRESS RESET IT  FILE NAME FILE TYPE ASCII FLAG FILE MODE TRANSFER ADDRESS START ADDRESS

Ø891	A65B	4C		INCA				MAKE FILE TYPE = 1	
Ø892			* ENTER H		R ASCII FI	LES			
Ø893	A65C	8E FF FF	LA65C	LDX	#\$FFFF			SET ASCII FLAG AND MODE = \$FF	
Ø894	A65F	ØD 78	LA65F	TST	FILSTA			IS FILE OPEN?	
Ø895 Ø896	A661 A663	26 B9 CE Ø1 DA		BNE LDU	LA61C #CASBUF			YES- FILE ALREADY OPEN CASSETTE INPUT BUFFER	
Ø897	A666	DF 7E		STU	CBUFAD			STORE IN STARTING ADDRESS	
Ø898	A668	A7 48		STA	8,U			FILE TYPE IN CASBUF+8	
Ø899	A66A	AF 49		STX	9,U			ASCII FLAG & MODE IN CASBUF+9, CASBUF+10	
0900			*	CASBU	F +8	+9	+10	TYPE ACCLE HOPE	
0901 0902			* BASIC (	PIINCHE	n		00	TYPE ASCII MODE 00	
Ø9Ø3			* BASIC /		,		טט	FF FF	
0904			* DATA					1 FF FF	
0905			* MACHINE			Ø2	ØØ	1	
Ø9Ø6			* MACHINI	E BLK L	DAD	02	ØØ	F	
Ø9Ø7 Ø9Ø8	A66C	8E Ø1 D2		LDX	#CFNBUF+	1		POINT X TO FILE NAME BUFFER	
0909	A66F	BD A5 98		JSR	LA598	-		MOVE 8 BYTES FROM (X) TO (U)	
0910	A672	ØF 7C		CLR	BLKTYP			ZERO BLOCK NUMBER	
0911	A674	86 ØF		LDA	#15			15 BYTES IN THE HEADER BLOCK	
Ø912	A676	97 7D BD A7 E5		STA	BLKLEN			CHAR COUNT	
Ø913 Ø914	A678 A67B	86 Ø2		JSR LDA	LA7E5 #2			GO WRITE ONE BLOCK OUTPUT FILE	
0915	A67D	97 78		STA	FILSTA			STORE IN FILE MODE	
Ø916	A67F	20 CF		BRA	LA65Ø			RESET POINTERS	
0917									
Ø918 Ø919	A681	05 01 04	* SEARCH LA681	FOR CO	RRECT CASS	ETTE FI	LE NAMI	CASSETTE BUFFER	
Ø919 Ø92Ø	A684	8E Ø1 DA 9F 7E	LAGSI	STX	#CASBUF CBUFAD			LOAD ADDRESS POINTER	
0921	A686	96 68	LA686	LDA	CURLIN			GET CURRENT LINE NUMBER MSB (CURLIN)	
0922	A688	4 C		INCA				IN DIRECT MODE IF ACCA = \$FF	
0923	A689	26 ØB		BNE	LA696			BRANCH IF NOT DIRECT MODE	
Ø924	A68B	BD A9 28		JSR	LA928			CLEAR SCREEN	
Ø925 Ø926	A68E A69Ø	9E 88 C6 53		LDX LDB	CURPOS #'S			CURRENT SCREEN CHAR POSITION S MEANS SEARCHING	
Ø927	A692	E7 81		STB	, X++			PUT AN S ON THE SCREEN	
Ø928	A694	9F 88		STX	CURPOS			STORE NEW CURSOR LOCATION	
0929	A696	8D 69	LA696	BSR	LA7Ø1			READ ONE BLOCK FROM TAPE	
Ø93Ø Ø931	A698	DA 7C		ORB	BLKTYP			OR ERROR FLAG WITH BLOCK NUMBER	
Ø932	A69A A69C	26 34 8E Ø1 DA		BNE LDX	LA6DØ #CASBUF			BRANCH IF NOT BLOCK ZERO OR ERROR POINT TO CASSETTE BUFFER	
Ø933	A69F	CE Ø1 D2		LDU	#CFNBUF+	1		POINT TO DESIRED NAME	
Ø934	A6A2	C6 Ø8		LDB	#8			EIGHT CHARACTERS MAX IN NAME	
0935	A6A4	6F E2		CLR	, - S			ZERO A BYTE ON THE STACK	
Ø936	A6A6	A6 8Ø 1Ø 9E 68	LA6A6	LDA LDY	,X+ CURLIN			GET CHAR FROM CASSETTE BLOCK GET CURLIN	
Ø937 Ø938	A6A8 A6AB	31 21		LEAY	1,Y			DIRECT MODE?	
Ø939	A6AD	26 Ø5		BNE	LA6B4			FALL THROUGH IF DIRECT MODE	
0940	A6AF	ØF 6F		CLR	DEVNUM			SET DEVICE NUMBER TO SCREEN	
0941	A6B1	BD A2 82		JSR	PUTCHR			OUTPUT A CHAR	
Ø942 Ø943	A6B4	AØ CØ	LA6B4	SUBA	, U+			SUBTRACT A CHAR FROM DESIRED NAME NON-ZERO RESULT IF NO MATCH	
Ø943	A6B6	AA E4		ORA	,\$			OR WITH TOP OF STACK, RESULT WILL BE NON	I-ZERO IE MISMATCH
Ø945	A6B8	A7 E4		STA	, S			SAVE IT	
Ø946	A6BA	5A		DECB				DONE ALL 8 CHARACTERS?	
Ø947	A6BB	26 E9		BNE	LA6A6			NO	
Ø948 Ø949	A6BD A6BF	A6 EØ 27 ØA		LDA BEQ	,S+ LA6CB			SEE IF ALL CHARS WERE OK BRANCH IF GOOD COMPARE	
0950	A6C1	6D 57		TST	-9,U			CHECK THE NUMBER OF CHARACTERS IN THE CLOA	D STATEMENT
0951	A6C3	27 Ø6		BEQ	LA6CB			IF NO NAME SPECIFIED, ANY FILE IS OK	
Ø952			* DIDN'T		HE RIGHT F	ILE IF	HERE		
Ø953 Ø954	A6C5 A6C7	8D ØA 26 Ø7		BSR BNE	LA6D1 LA6DØ			LOOK FOR FILE RETURN IF ERROR	
Ø955		20 BB		BRA	LAGDU LAGBG			GO LOOK SOME MORE	
Ø956		86 46	LA6CB	LDA	#'F			*	
Ø957		8D 29		BSR	LA6F8			* PUT F ON THE SCREEN IF DIRECT MODE	
0958	A6CF			CLRA				SET ZERO FLAG TO INDICATE NO ERRORS	
Ø959 Ø96Ø	A6DØ A6D1	39 7D Ø1 E4	LA6DØ LA6D1	RTS TST	CASBUF+1	α		CHECK FILE MODE	
Ø961		26 Ø9	LAUDI	BNE	LA6DF	U		BRANCH IF ASCII OR DATA	
0962		BD A7 7C		JSR	CASON			TURN ON TAPE DECK	
Ø963		8D 3Ø	LA6D9	BSR	GETBLK			LOAD A BLOCK FROM TAPE	
0964		8D Ø8		BSR	LA6E5			CHECK FOR ERROR OR LAST BLOCK	
Ø965 Ø966		20 FA 8D 20	LA6DF	BRA BSR	LA6D9 LA7Ø1			KEEP GOING READ ONE BLOCK FROM TAPE	
Ø967		8D Ø2	LAUDI	BSR	LAGE5			CHECK FOR ERROR OR LAST BLOCK	
Ø968		20 FA		BRA	LA6DF			KEEP READING BLOCKS	
Ø969		26 Ø6	LA6E5	BNE	LA6ED			GOT AN ERROR ON READING IN BLOCK	
Ø97Ø		96 7C		LDA	BLKTYP			GET BLOCK NUMBER	
Ø971 Ø972	A6E9	40 2B 14		NEGA BMI	LA700			CHECK FOR LAST BLOCK RETURN IF NOT AN END OF PROGRAM BLOCK	
Ø972 Ø973	A6EC			DECA	EU/ 22			IF BLOCK NUMBER WAS \$FF, ACCA IS NOW ZERO	- THIS WILL
Ø974			*					CAUSE CLOAD TO IGNORE ERRORS IN THE	
Ø975			*					BLOCKS WHICH IT IS SKIPPING WHILE	
Ø976	4655	07.01	*	CT.	000555			LOOKING FOR THE CORRECT FILE NAME.	
Ø977 Ø978		97 81 32 62	LAGED	STA LEAS	CSRERR 2 S			STORE ACCA TO ERROR FLAG REMOVE RETURN ADDRESS FROM STACK	
Ø979		20 12		BRA	2,3 LA7Ø5			TURN OFF MOTOR	

Ø98Ø	A6F3	B6 Ø4 Ø	D LA6F3	LDA	VIDRAM	CET EIDST CHAD ON SCREEN
Ø981	A6F6	88 4Ø	LAUFS	EORA	#\$4Ø	GET FIRST CHAR ON SCREEN
Ø982	A6F8	D6 68	LA6F8	LDB	CURLIN	REVERSE THE VIDEO GET CURLIN MSB
Ø983	A6FA	5C	LAUFO	INCB	CORLIN	CHECK FOR DIRECT MODE
					1 4 7 9 9	
Ø984	A6FB	26 Ø3	1	BNE	LA700	BRANCH IF NOT DIRECT MODE
Ø985	A6FD	B7 Ø4 Ø		STA	VIDRAM	PUT IT ON SCREEN
Ø986	A7ØØ	39	LA700	RTS		
Ø987			+ 0540	A DI 001/	FROM CACCETTE	
Ø988	4701	00 70			FROM CASSETTE	CTART TARE, AND LOOK FOR A RUNCH OF ASS OR ANA RVIES
Ø989	A7Ø1	8D 79	LA7Ø1	BSR	CASON	START TAPE, AND LOOK FOR A BUNCH OF \$55 OR \$AA BYTES
0990	A7Ø3	8D Ø6		BSR	GETBLK	READ A BLOCK
Ø991	A7Ø5	BD A7 E	EA7Ø5	JSR	LA7E9	TURN OFF MOTOR
Ø992	A7Ø8	D6 81		LDB	CSRERR	GET ERROR STATUS
Ø993	A7ØA	39		RTS		
Ø994	A7ØB	1A 5Ø	GETBLK	ORCC	#\$50	DISABLE IRQ,FIRQ
Ø995	A7ØD	8D E4		BSR	LA6F3	REVERSE VIDEO UPPER LEFT CHAR IF DIRECT MODE
Ø996	A7ØF	9E 7E		LDX	CBUFAD	GET LOAD ADDRESS
Ø997	A711	4 F		CLRA		RESET ACCA
Ø998	A712	8D 41	LA712	BSR	LA755	READ A BIT FROM TAPE, RETURN IT IN CARRY FLAG
Ø999	A714	46		RORA		PUT BIT IN MSB OF ACCA
1000	A715	81 3C		CMPA	#\$3C	GET SYNC ED ON \$3C
1001	A717	26 F9		BNE	LA712	NOT SYNC ED YET
1002	A719			BSR	LA749	GET BLOCK NUMBER
1003	A71B	97 7C		STA	BLKTYP	SAVE IT
1004	A71D	8D 2A		BSR	LA749	GET CHAR COUNT
1005	A71F	97 7D		STA	BLKLEN	SAVE IT
1006		9B 7C			BLKTYP	ACCUMULATE CHECKSUM
1007	A721	97 8Ø		STA	CCKSUM	SAVE IT
1007	A725	96 7D		LDA	BLKLEN	GET BACK CHAR COUNT
1008	A725	96 70		STA	CSRERR	TEMP SAVE
1010	A729	27 10	1.4700	BEQ	LA73B	NULL SET OF CHARACTERS
1011	A72B	8D 1C A7 84	LA72B	BSR	LA749	GET BYTE FROM TAPE
1012	A72D			STA	, X	FILL MEMORY WITH TAPE DATA
1013	A72F	A1 8Ø		CMPA	, X+	SEE IF WE READ BACK SAME THING
1014	A731	26 11		BNE	LA744	BRANCH IF NOT PUTTING IT IN RAM
1015	A733	9B 8Ø		ADDA	CCKSUM	ACCUMULATE CHECKSUM
1016		97 8Ø		STA	CCKSUM	TEMP STORE CHECKSUM
1017	A737	ØA 81		DEC	CSRERR	DECR TEMP CHAR COUNT
1018	A739	26 FØ		BNE	LA72B	GET ANOTHER CHARACTER
1019	A73B	8D ØC	LA73B	BSR	LA749	GET CHECKSUM FROM TAPE
1020	A73D	90 80		SUBA	CCKSUM	COMPARE TO CALCULATED CHECKSUM
1021	A73F	27 Ø5		BEQ	LA746	BRANCH IF OK
1022	A741	86 Ø1		LDA	#1	CHECKSUM ERROR FLAG
1023	A743	8C		FCB	SKP2	SKIP TWO BYTES
1024	A744	86 Ø2	LA744	LDA	#2	NON-RAM ERROR FLAG
1025	A746	97 81	LA746	STA	CSRERR	1 IF CHECKSUM ERROR, 2 IF LOADING INTO NON-RAM
1026	A748	39		RTS		
1027						
1028			* GET A	A BYTE FR	OM TAPE	
1029	A749	86 Ø8	LA749	LDA	#8	8 BITS/BYTE
1030	A74B	97 82		STA	CPULWD	TEMP COUNTER
1031	A74D	8D Ø6	LA74D	BSR	LA755	READ A BIT FROM TAPE
1031	A74F	46	CA7 40	RORA	ER7 55	PUT IT INTO ACCA
1032	A75Ø	ØA 82		DEC	CPULWD	GOT ALL 8 BITS
1034	A752	26 F9		BNE	LA74D	NO
1035	A754	39		RTS	ER/ 40	no
1036	A7 54	33		KIJ		
1037			* DEAD	A DIT ED	OM THE TAPE	
1037	A755	8D Ø6	LA755	BSR	LA75D	GET THE TIME BETWEEN TRANSITIONS
1038	A755	D6 83	LM/33	LDB	CPERTM	* GET PERIOD TIMER
1039	A757	5A		DECB	OI ERIH	* GET PERIOD TIMER
					CMPMID	CMPMID CONTAINS 18 INITIALLY, AND IS USED TO DETERMINE
1041 1042	M/ DA	D1 8F		CMFB	CMPMID	WHETHER THE BIT READ IS A ONE OR ZERO
			*			
1043			*			IF THE PERIOD TIMER IS < 18, THE BIT
1044	4750	20	*	D.T.C		IS CONSIDERED TO BE A ONE, IF > 18, IT IS ZERO
1045	A75C	39		RTS		
1046						
1047				TIMING L		
1048		ØF 83	LA75D	CLR	CPERTM	RESET PERIOD TIMER
1049		ØD 84		TST	CBTPHA	CHECK TO SEE IF SYNC ED ON THE HI-LO TRANSITION OR LO-HI
1050	A761	26 10		BNE	LA773	BRANCH ON HI-LO TRANSITION
1051				HI TRANS		
1052		8D Ø7	LA763	BSR	LA76C	READ CASSETTE INPUT BIT
1053	A765	25 FC		BCS	LA763	LOOP UNTIL IT IS LO
1054	A767	8D Ø3	LA767	BSR	LA76C	READ CASSETTE INPUT DATA
1055	A769	24 FC		BCC	LA767	WAIT UNTIL IT GOES HI
1056	A76B	39		RTS		
1057						
1058			* READ	CASSETTE	INPUT BIT OF THE PIA	
1059	A76C	ØC 83	LA76C	INC	CPERTM	INCREMENT PERIOD TIMER
1060		F6 FF 2		LDB	PIA1	GET CASSETTE INPUT BIT
1061	A771			RORB		PUT CASSETTE BIT INTO THE CARRY FLAG
1062	A771			RTS		
1062	n//4	55		K13		
1063			<b>↓</b> MATT	EUD HI	LO TRANSITION	
	A772	QD 57	* WAII LA773	BSR -	LA76C	DEAD CASSETTE INDUT DATA
1065		8D F7	LA//3			READ CASSETTE INPUT DATA
1066		24 FC		BCC	LA773	LOOP UNTIL IT IS HI
1067	A777	8D F3	LA777	BSR	LA76C	READ CASSETTE INPUT
1068		25 FC		BCS	LA777	LOOP UNTIL IT IS LO

1069	A77B	39			RTS		
1070 1071				***	FOR THE	SYNC BYTES - RETURN WITH	$ACCA = \emptyset$ IF SYNC FD
1072						TRANSITION, ACCA = \$AØ IF S	
1073						SITION OF THE INPUT SIGNAL	
1074		1A 5		CASON	ORCC	#\$50	DISABLE IRQ,FIRQ
1075	A77E	8D 4			BSR	LA7CA	TURN ON TAPE DECK MOTOR
1076 1077	A78Ø A782	ØF 8 8D D		LA782	CLR BSR	CPULWD LA763	RESET UP TO SPEED COUNTER WAIT FOR LO-HI TRANSITION
1077	A784	8D 2		LA784	BSR	LA7AD	WAIT FOR LO-HI TRANSITION WAIT FOR HI-LO TRANSITION
1079	A786	22 Ø			BHI	LA797	CASSETTE SPEED IN RANGE FOR 1200 HZ
1080	A788	8D 1	D	LA788	BSR	LA7A7	WAIT FOR LO-HI TRANSITION
1081	A78A	25 Ø			BCS	LA79B	CASSETTE SPEED IN RANGE FOR 2400 HZ
1082	A78C	ØA 8			DEC	CPULWD	DECREMENT UP TO SPEED COUNTER IF SYNC ED ON LO-HI
1083 1084	A78E A79Ø	96 8 81 A			LDA CMPA	CPULWD #-96	GET IT HAVE THERE BEEN 96 CONSECUTIVE 1-0-1-0 PATTERNS
1085	A792	26 E		LA792	BNE	LA782	NO
1086	A794	97 8			STA	СВТРНА	SAVE WHICH TRANSITION (HI-LO OR LO-HI)
1087	A796	39			RTS		
1088	A797	8D Ø		LA797	BSR	LA7A7	WAIT FOR LO-HI TRANSITION
1089	A799	22 E		1.4700	BHI	LA784	BRANCH IF TWO CONSECUTIVE 1200 HZ PULSES
1090 1091	A79B A79D	8D 1 25 E		LA79B	BSR BCS	LA7AD LA788	WAIT FOR HI-LO TRANSITION BRANCH IF TWO CONSECUTIVE 2400 HZ PULSES
1092	A79F	ØC 8			INC	CPULWD	INCREMENT UP TO SPEED COUNTER IF SYNC ED ON HI-LO
1093	A7A1	96 8			LDA	CPULWD	GET IT
1094	A7A3	8Ø 6	Ø		SUBA	#96	GOT ENOUGH SYNC PULSES? - ACCA WILL BE ZERO IF
1095				*			THERE HAVE BEEN 96 CONSECUTIVE Ø-1-Ø-1 PATTERNS
1096	A7A5	2Ø E		1 4 7 4 7	BRA	LA792 CPERTM	DECET DEDIOD TIMED
1097 1098	A7A7 A7A9	ØF 8 8D B		LA7A7	CLR BSR	LA767	RESET PERIOD TIMER WAIT UNTIL CASSETTE INPUT GOES HI
1099	A7AB	2Ø Ø			BRA	LA781	WALL DIVILE CASSELLE INFOL GOES HI
1100	A7AD	ØF 8		LA7AD	CLR	CPERTM	RESET PERIOD TIMER
1101	A7AF	8D C	6		BSR	LA777	WAIT UNTIL CASSETTE GOES LO
1102	A7B1	D6 8		LA7B1	LDB	CPERTM	GET PERIOD TIMER
1103	A7B3	D1 9			CMPB	CMPØ	UPPER LIMIT OF 1200 HZ PERIOD
1104 1105	A7B5 A7B7	22 Ø D1 9			BHI CMPB	LA7BA CMP1	BRANCH IF CASSETTE SPEED IS TOO SLOW OR DROPOUT UPPER LIMIT OF 2400 HZ PERIOD
1105	A7B9	39	1		RTS	CHFI	OFFER LIMIT OF 2400 HZ FERIOD
1107	A7BA	ØF 8	2	LA7BA	CLR	CPULWD	RESET UP TO SPEED COUNTER
1108	A7BC	39			RTS		
1109							
1110				* MOTOR			
1111 1112	A7BD A7BF	1F 8 9D 9		MOTOR	TFR JSR	A,B GETNCH	SAVE CURRENT TOKEN IN ACCB GET NEXT INPUT CHARACTER FROM BASIC
1112	A7C1	C1 A			CMPB	#\$AA	OFF TOKEN
1114		27 2			BEQ	LA7E9	YES
1115	A7C5	C1 8	8		CMPB	#\$88	ON TOKEN
1116	A7C7	BD A	5 C9		JSR	LA5C9	SYNTAX ERROR IF IT WASN T ON OR OFF
1117		B6 F		LA7CA	LDA	PIA1+1	READ CRA OF U4
1118 1119	A7CD A7CF	8A Ø 8D 1			ORA BSR	#\$08	TURN ON BIT 3 WHICH ENABLES MOTOR DELAY PUT IT BACK
1119	A7D1	9E 8		LA7D1	LDX	LA7FØ ZERO	GET READY TO WAIT A WHILE
1121	N/01	JL 0	n	LATUI	LDX	ZENO	del Readi 10 Mail A Milee
1122				* DELAY W	HILE DE	CREMENTING X TO ZERO	
1123	A7D3	30 1		LA7D3	LEAX	-1,X	DECREMENT X
1124	A7D5	26 F	С		BNE	LA7D3	BRANCH IF NOT ZERO
1125 1126	A7D7	39			RTS		
1127				* SEND SY	NCLN \$5	55 S TO TAPE	
1128	A7D8	1A 5	Ø	WRLDR	ORCC		DISABLE INTERRUPTS
1129	A7DA	8D E			BSR	LA7CA	TURN ON TAPE DECK MOTOR
1130	A7DC				LDX	SYNCLN	GET COUNT OF \$55 S TO SEND
1131	A7DE			LA7DE		LA828	SEND \$55 TO TAPE ARE ALL \$55 S SENT?
1132 1133	A7EØ A7E2				BNE	-1,X LA7DE	NO
1134	A7E4				RTS	277.02	
1135							
1136						TES AND A BLOCK TO TAPE	
1137	A7E5			LA7E5		WRLDR	WRITE SYNC BYTES TO TAPE
1138	A7E7	8D Ø	В	LA7E7	BSR	SNDBLK	GO WRITE A BLOCK
1139 1140				* TIIDN OF	E TADE	DECK MOTOR	
1141	A7E9	1C A	F	LA7F9	ANDCC		ENABLE IRQ,FIRQ
1142	A7EB					PIA1+1	READ CRA OF U4
1143	A7EE					#\$F7	TURN OFF BIT 3
1144	A7FØ		F 21	LA7FØ		PIA1+1	PUT IT BACK
1145	A7F3	39			RTS		
1146 1147				* WDITE A	BLOCK	TO CASSETTE	
1148				* BUFFER			
1149						IN CBUFAD	
1150				* BLOCK N			
1151	A7F4			SNDBLK	ORCC		DISABLE IRQ, FIRQ
1152	A7F6					BLKLEN	GET CHAR COUNT
1153 1154	A7F8 A7FA					CSRERR BLKLEN	TEMP CHAR COUNT GET CHAR COUNT (INCLUDED IN CHECKSUM)
1154	A7FC					LA8Ø5	BRANCH IF NO CHARACTERS - NULL
1156	A7FE				LDX	CBUFAD	GET STARTING ADDRESS
1157	A8ØØ			LA8ØØ	ADDA		CHECKSUM THE BUFFER

1158	A8Ø2	5 A					DECB		DONE ALL CHARACTERS?
1159		26 FB					BNE	LA800	NO
1160		9B 7C				LA8Ø5		BLKTYP	ADD IN THE BLOCK NUMBER
1161	A8Ø7	97 80					STA	CCKSUM	SAVE THE CHECKSUM
1162	A8Ø9	9E 7E					LDX	CBUFAD	GET STARTING ADDRESS
1163	A8ØB						BSR	LA828	SEND \$55 TO TAPE
1164		86 3C					LDA	#\$3C	SYNC CHAR
1165		8D 19					BSR	LA82A	SEND TO TAPE
1166	A811						LDA	BLKTYP	GET BLOCK NUMBER
1167 1168	A815	8D 15					BSR LDA	LA82A BLKLEN	SEND BLOCK NUMBER TO TAPE GET CHARACTER COUNT
1169	A817						BSR	LA82A	SEND CHAR COUNT TO TAPE
		4D					TSTA	ENGEN	SET FLAGS
1171		27 Ø8					BEQ	LA824	BRANCH IF CHAR COUNT IS ZERO
1172	A81C	A6 8Ø				LA810	LDA	, X+	GET BUFFER CHARACTER
1173	A81E	8D ØA					BSR	LA82A	SEND BUFFER TO TAPE
1174	A82Ø						DEC	CSRERR	DECR TEMP CHAR COUNT
1175		26 F8					BNE	LA81C	NOT DONE YET
1176		96 80				LA824		CCKSUM	GET CHECKSUM
1177		8D Ø2				1 4020	BSR LDA	LA82A #\$55	SEND CHECKSUM TO TAPE
1178 1179	A626	86 55				LA828	LDA	#\$55	SEND A \$55 TO TAPE
1180						* THI	S ROUTINE	SENDS THE A REG TO TAPE	
1181	A82A	34 Ø2				LA82A		A	SAVE OUTPUT CHARACTER
1182		C6 Ø1					LDB	#1	ACCB CONTAINS A MASK USED TO DETERMINE WHETHER A
1183						*			BIT IN THE OUTPUT CHARACTER IS HI OR LO
1184	A82E	96 85				LA82E	LDA	CLSTSN	GET THE ENDING VALUE OF THE LAST SINE CYCLE
1185		B7 FF					STA	DA	STORE IN THE D/A CONVERTER
1186		10 8E	8A	5 C			LDY	#LA85C	SINE LOOK-UP TABLE FOR GENERATING FSK
1187		E5 E4					BITB	, S	IS THE CURRENT BIT A ONE OR A ZERO ?
1188	A839	26 ØD					BNE	LA848	IF A 1, DO HIGH FREQ
1189 1190	402D	A6 AØ				* LOW LA83E	FREQUENCY		USE EVERY BYTE IN TABLE IF LOW FREQUENCY
		10 8C	ΔΩ	Rα		LAGSE		,Y+ #LA85C+36	END OF SINE TABLE?
1192		27 12	АО	Oυ			BEQ	LA855	YES
1193		B7 FF	20				STA	DA	SEND NEXT VALUE TO D/A CONVERTER
1194	A846						BRA	LA83B	GET NEXT VALUE
1195						* HIG	H FREQUENCY	Y LOOK UP	
1196	A848	A6 A1				LA848	LDA	,Y++	USE EVERY OTHER BYTE IF HIGH FREQUENCY
1197		10 8C	8A	80			CMPY		END OF SINE TABLE?
	A84E						BEQ	LA855	YES
1199		B7 FF	20				STA	DA	SEND NEXT VALUE TO D/A CONVERTER
1200		2Ø F3 97 85				LA855	BRA	LA848	GET NEXT VALUE
1201 1202	A857	58				LABSS	STA ASLB	CLSTSN	SAVE THE LAST VALUE SENT TO THE D/A CONVERTER SHIFT MASK BIT LEFT
		30							SHITT HASK BIT ELLT
1203	A858	24 D4					BCC	LA82E	DONE WHEN MASK BIT IS SHIFTED INTO CARRY FLAG
1203 1204		24 D4 35 82					BCC PULS	LA82E A.PC	DONE WHEN MASK BIT IS SHIFTED INTO CARRY FLAG RESTORE OUTPUT CHARACTER AND RETURN
1203 1204 1205		24 D4 35 82					BCC PULS		DONE WHEN MASK BIT IS SHIFTED INTO CARRY FLAG RESTORE OUTPUT CHARACTER AND RETURN
1204						* THI	PULS		RESTORE OUTPUT CHARACTER AND RETURN
1204 1205							PULS S IS A LOOF T 1 IS USEF	A,PC K-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208	A85A A85C	35 82 82 92				* (BI A LA850	PULS S IS A LOOP T 1 IS USER	A,PC <-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209	A85A A85C A862	35 82 82 92 EA F2	FA	FA	FA F	* (BI A LA850 2	PULS S IS A LOOI T 1 IS USEI FCB FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA,\$F	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209 1210	A85A A85C A862 A868	35 82 82 92 EA F2 EA DA	FA CA	FA BA	FA F AA 9	* (BI A LA850 2 2	PULS S IS A LOOI T 1 IS USEI FCB FCB FCB	A,PC  K-UP TABLE OF SINE VALUES TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209 1210 1211	A85A A85C A862 A868 A86E	35 82 82 92 EA F2 EA DA 7A 6A	FA CA 52	FA BA 42	FA F AA 9 32 2	* (BI A LA850 2 2 2	PULS S IS A LOOK T 1 IS USER FCB FCB FCB FCB	A,PC  (-UP TABLE OF SINE VALUES ) TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,32,\$22	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209 1210 1211 1212	A85C A862 A868 A86E A874	35 82 82 92 EA F2 EA DA 7A 6A 12 ØA	FA CA 52 Ø2	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A	PULS S IS A LOOK T 1 IS USER FCB FCB FCB FCB FCB	A,PC  (-UP TABLE OF SINE VALUES ) TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$0A,\$02,\$02,\$02,\$02	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213	A85C A862 A868 A86E A874	35 82 82 92 EA F2 EA DA 7A 6A	FA CA 52 Ø2	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A	PULS S IS A LOOK T 1 IS USER FCB FCB FCB FCB	A,PC  (-UP TABLE OF SINE VALUES ) TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$0A,\$02,\$02,\$02,\$04	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209 1210 1211 1212	A85C A862 A868 A86E A874	35 82 82 92 EA F2 EA DA 7A 6A 12 ØA	FA CA 52 Ø2	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A	PULS S IS A LOOI T 1 IS USEI FCB FCB FCB FCB FCB FCB FCB FCB	A,PC  (-UP TABLE OF SINE VALUES ) TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$0A,\$02,\$02,\$02,\$02	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214	A85C A862 A868 A86E A874 A87A	35 82 82 92 EA F2 EA DA 7A 6A 12 ØA	FA CA 52 Ø2	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB FCB FCB FCB FCB FCB FCB FCB	A,PC  (-UP TABLE OF SINE VALUES ) TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$0A,\$02,\$02,\$02,\$02	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217	A85A A85C A862 A868 A86E A874 A87A	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$04,\$04,\$02,\$02,\$02,\$04 \$12,\$22,\$32,\$42,\$52,\$6A	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218	A85A A85C A862 A868 A86E A87A A87A	82 92 EA F2 EA DA 7A 6A 12 0/2 22 8D 3F 34 10/8D B7	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F4,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$03 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219	A85A A85C A862 A868 A86E A874 A87A A88Ø A882 A884 A887	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø BD B7 35 1Ø	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES ) TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$04 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220	A85A  A85C A862 A868 A874 A87A  A880 A882 A884 A887	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø BD B7 35 1Ø C1 Ø8	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$04 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1219	A85A  A85C A862 A868 A874 A87A  A880 A882 A884 A887 A889 A888	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø BD B7 35 1Ø 80 22 48	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES ) TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$04 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222	A85A  A85C A862 A868 A86E A874 A87A  A88Ø A882 A884 A887 A888 A88B	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø BD 87 35 1Ø C1 Ø8 54 85 A	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$03 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM 0-8 TO (-1 TO 7)
1204 1205 1206 1207 1208 1209 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222	A85A  A85C A862 A868 A864 A87A  A880 A882 A884 A887 A889 A888 A88B A88B	82 92 EA F2 EA DA 7A 6A 12 02 8D 3F 34 10 BD 87 35 10 C1 08 22 48 5A 2B 05	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$232,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5 LA895	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø)
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222	A85A  A85C A862 A868 A864 A87A  A880 A882 A884 A887 A889 A888 A88B A88B	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø BD B7 35 1Ø C1 Ø8 22 48 5A 28 Ø5 86 1Ø	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$03 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM 0-8 TO (-1 TO 7)
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224	A85A  A85C A862 A868 A86E A874 A87A  A880 A882 A884 A887 A889 A88B A88D A88B A890 A892	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø BD B7 35 1Ø C1 Ø8 22 48 5A 28 Ø5 86 1Ø	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$232,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5 LA895	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IT SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224	A85A  A85C A862 A868 A868 A874 A87A  A880 A882 A884 A887 A888 A880 A888 A880 A888 A890 A888	82 92 EA F2 EA DA 7A 6A 12 ØA 12 22 8D 3F 34 1Ø 8D 87 35 1Ø C1 Ø8 22 48 5A 28 Ø5 86 1Ø 30	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA850 2 2 2 2 A A	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5 LA895 #*10	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1229 1220 1221 1222 1223 1224 1225 1226 1227 1227 1228	A85A  A85C A862 A868 A86E A874 A87A  A88Ø A882 A884 A887 A889 A88B A88B A88B A88B A89Ø A892 A893	82 92 EA F2 EA DA 7A 6A 12 0A 12 22 8D 3F 34 10 8D 87 35 10 C1 08 22 48 5A 28 05 86 10 30 08	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85G 2 2 2 2 A A * SET SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES  TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA \$EA,\$PA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$232,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5  LA895 #\$10  LA89D	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC
1204 1205 1206 1207 1208 1209 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227	A85A A85C A862 A868 A86E A874 A87A A888 A884 A887 A889 A888 A892 A893 A895 A893 A895 A897 A899	82 92 EA F2 EA DA 7A 6A 12 0A 12 22 8D 3F 35 10 C1 08 22 48 5A 2B 05 86 10 3D 20 08 E6 84 27 08 3C4 70	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85G 2 2 2 2 A A * SET SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$22,\$32,\$42,\$52,\$6A  LABC1 X LB738 X #8 LAB05 LA895 #*10 LA890 ,X LA890 ,X LA890 ,X LA890 #70	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  GET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$1Ø OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO
1204 1205 1206 1207 1208 1210 1211 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1228 1228	A85A  A85C  A862  A868  A867  A87A  A880  A889  A889  A889  A899  A899  A899  A899	82 92 EA FZ EA DA 7A 6A 12 ØA 12 22  8D 3F 34 1Ø BD 87 35 1Ø 80 05 86 1Ø 3D 20 Ø8 E6 84 2A Ø3 C4 7Ø	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$23,\$42,\$52,\$6A  LABC1 X LB738 X H8 LA8D5 LA895 #\$10  LA89D ,X LA89D	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1227 1228 1229 1229 1221	A85A  A85C  A862  A868  A868  A87A  A880  A889  A889  A889  A899  A899  A899  A899  A899  A899	82 92 EA F2 EA DA 7A 6A 12 0A 12 22  8D 3F 34 10 BD 87 35 10 C1 08 22 48 5A 05 86 10 3D 20 08 42 24 03 C4 70 21 5F	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85G 2 2 2 2 A A * SET SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$23,\$42,\$52,\$6A  LABC1 X LB738 X #8 LABD5 LA895 #\$10 LA895 #\$10 LA890 ,X LA89C #\$70 SKP1	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1230 1231 1231	A85A A85C A862 A868 A868 A874 A87A A888 A880 A888 A880 A889 A897 A899 A898 A897 A899 A898 A897 A899 A898	82 92 EA F2 EA DA 7A 6A 12 0A 12 22  8D 3F 35 10 C1 08 22 48 5A 05 86 10 30 C2 08 86 84 20 08 86 70 30 C4 70 21 57 34 04	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FF,\$FF,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$20A,\$02,\$02,\$02,\$04 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5 LA895 #\$10  LA890 ,X LA890 ,X LA890 ,X LA890 ,X LA890 #\$70  SKP1 B	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO
1204 1205 1206 1207 1208 1210 1211 1211 1212 1213 1214 1215 1216 1217 1218 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1229 1230 1231 1231 1231 1232 1233 1234 1231 1232 1233 1234 1231 1232 1233 1234 1231 1231	A85A  A85C  A862  A868  A87A  A880  A884  A884  A888  A889  A889  A899  A899  A899  A897  A899  A897  A899  A897	82 92 EA PA 7A 6A 12 ØA 12 22  8D 3F 34 1Ø BD 87 35 1Ø 60 08 86 1Ø 3D 20 Ø8 E6 84 2A Ø3 C4 7Ø 21 5F 34 Ø4 6C	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02,\$0A \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5 LA895 #\$10  LA895 #\$10  LA890 ,X LA89C #\$70 SKP1  B LA90D	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR )
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1222 1222 1222 1222	A85A  A85C  A862  A868  A868  A87A  A887  A889  A889  A899  A899  A899  A899  A899  A899  A898  A890  A898	82 92 EA F2 EA PA 7A 6A 12 0A 12 22  8D 3F 34 10 BD 87 35 10 C1 08 22 48 5A 05 86 10 3D 0 86 84 2A 03 C4 70 21 5F 34 04 8A 04 8A 04 8A 04	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FF,\$FF,\$FA,\$F2 \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$20A,\$02,\$02,\$02,\$04 \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5 LA895 #\$10  LA890 ,X LA890 ,X LA890 ,X LA890 ,X LA890 #\$70  SKP1 B	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X, Y, Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN
1204 1205 1206 1207 1208 1210 1211 1211 1212 1213 1214 1215 1216 1217 1218 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1229 1230 1231 1231 1231 1232 1233 1234 1231 1232 1233 1234 1231 1232 1233 1234 1231 1231	A85A  A85C  A862  A868  A868  A87A  A887  A889  A889  A899  A899  A899  A899  A899  A899  A898  A890  A898	82 92 EA F2 EA DA 7A 6A 12 0A 12 22  80 3F 80 3F 81 10 80 87 35 10 C1 08 82 48 5A 28 05 86 10 30 08 E6 84 2A 03 C4 70 21 34 04 8D 6C A6 84 66 84	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02 \$12,\$232,\$42,\$52,\$6A  LABC1 X LB738 X #8 LABD5  LA895 #\$10  LA895 #\$70  SKP1  B LA90D ,X	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR )
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1229 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1233 1233 1233	A85A  A85C  A862  A868  A87A  A880  A884  A884  A888  A889  A889  A899  A899  A897  A899  A897  A899  A897  A897  A897	82 92 EA F2 EA DA 7A 6A 12 0A 12 22  80 3F 80 3F 81 10 80 87 35 10 C1 08 82 48 5A 28 05 86 10 30 08 E6 84 2A 03 C4 70 21 34 04 8D 6C A6 84 66 84	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$CA,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC
1204 1205 1206 1207 1208 1210 1211 1211 1212 1213 1214 1215 1216 1217 1218 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1232 1233 1234 1235 1236 1237 1238	A85A  A85C  A862  A868  A868  A87A  A889  A889  A889  A899  A899  A899  A899  A899  A899  A890	82 92 EA PA 7A 6A 12 ØA 12 22  8D 3F 34 1Ø C1 Ø8 22 48 55 46 86 1Ø 3D Ø5 86 1Ø 3D Ø6 84 2A Ø3 C4 7Ø 21 5F 48 80 6C A6 84 2B Ø1 4F	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET LA895	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$CA,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF
1204 1206 1207 1208 1210 1211 1211 1212 1213 1214 1215 1216 1217 1218 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1238 1238 1238	A85A  A85C  A862  A868  A87A  A888  A888  A888  A888  A898  A899  A891  A897  A899  A897  A899  A897  A849  A848  A848  A848  A848  A848	82 92 EAA DA 7A 6A 12 0A 12 0A 8D 3F 34 10 C1 08 85 86 10 30 08 E6 84 2A 03 C4 70 21 5F 04 04 8D 6C A6 84 2B 05 A6 84 A7 04 A7 04 A8 05 A6 84 A7 05 A7	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET LA895 LA896	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02,\$0A \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5  LA8D5  LA895 #\$10  LA895 #\$10  LA890 ,X LA890 ,X LA896 #\$70  SKP1  B LA90D ,X LA846 #\$0F GRBLOK ,S+	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF SAVE ONLY PIXEL ON/OFF INFO OR WITH WHICH PIXEL TO TURN ON OR IN THE COLOR
1204 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1231 1232 1233 1234 1235 1236 1237 1238 1238 1238 1238 1238 1238 1238 1238	A85A  A85C  A862  A868  A868  A87A  A880  A889  A889  A889  A889  A899  A891  A897  A897  A897  A897  A897  A898  A89A  A89A  A89A  A89A  A89A  A89A  A89A  A89A	82 92 EA FZ EA DA 7A 6A 12 0A 12 22  8D 3F 34 10 BD 87 35 10 80 08 22 48 5A 05 86 10 3D 08 E6 84 2A 03 C4 70 21 5F 34 06 A6 84 2B 01 4F 84 0F 9A 86 AA E0 AA 80	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET LA895	PULS  S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$CA,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF SAVE ONLY PIXEL ON/OFF INFO OR WITH WHICH PIXEL TO TURN ON OR IN THE COLOR FORCE GRAPHIC
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1227 1228 1229 1230 1231 1232 1238 1234 1235 1236 1237 1238 1237 1238 1237 1238 1237 1238 1237 1238 1237 1238 1237 1238 1237 1238	A85A  A85C  A862  A868  A868  A868  A887  A889  A889  A889  A899  A899  A899  A891  A890  A891  A891  A890  A891  A894  A896  A896  A896  A897  A898  A896  A896  A897  A898  A896  A896  A897  A898  A896  A897  A898  A896  A897  A898  A897  A898	82 92 EA F2 EA DA 7A 6A 12 02  80 3F 34 10 BD 87 35 10 C1 08 22 48 63 05 86 10 30 08 22 48 24 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 30 68 84 28 01 4F 9A 86 AA 80 AA 80 AA 80 AA 84	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET LA895 LA896	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$02,\$02,\$02,\$02,\$0A \$12,\$22,\$32,\$42,\$52,\$6A  LA8C1 X LB738 X #8 LA8D5  LA8D5  LA895 #\$10  LA895 #\$10  LA890 ,X LA890 ,X LA896 #\$70  SKP1  B LA90D ,X LA846 #\$0F GRBLOK ,S+	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NINE ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF SAVE ONLY PIXEL ON/OFF INFO OR WITH WHICH PIXEL TO TURN ON OR IN THE COLOR
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1220 1221 1223 1224 1225 1226 1227 1228 1230 1231 1232 1233 1234 1235 1236 1237 1238 1238 1239 1240 1241	A85A  A85C  A862  A868  A868  A87A  A880  A889  A889  A889  A889  A899  A891  A897  A897  A897  A897  A897  A898  A89A  A89A  A89A  A89A  A89A  A89A  A89A  A89A	82 92 EA F2 EA DA 7A 6A 12 02  80 3F 34 10 BD 87 35 10 C1 08 22 48 63 05 86 10 30 08 22 48 24 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 30 68 84 28 01 4F 9A 86 AA 80 AA 80 AA 80 AA 84	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET LA895 LA896	PULS  S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$CA,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF SAVE ONLY PIXEL ON/OFF INFO OR WITH WHICH PIXEL TO TURN ON OR IN THE COLOR FORCE GRAPHIC
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1226 1227 1228 1230 1231 1232 1233 1234 1235 1236 1237 1238 1238 1239 1240 1241 1242	A85A  A85C  A862  A868  A868  A868  A887  A889  A889  A889  A899  A899  A899  A891  A890  A891  A891  A890  A891  A894  A896  A896  A896  A897  A898  A896  A896  A897  A898  A896  A896  A897  A898  A896  A897  A898  A896  A897  A898  A897  A898	82 92 EA F2 EA DA 7A 6A 12 02  80 3F 34 10 BD 87 35 10 C1 08 22 48 63 05 86 10 30 08 22 48 24 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 30 68 84 28 01 4F 9A 86 AA 80 AA 80 AA 80 AA 84	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET SET	PULS S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$CA,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF SAVE ONLY PIXEL ON/OFF INFO OR WITH WHICH PIXEL TO TURN ON OR IN THE COLOR FORCE GRAPHIC
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1222 1222 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1231 1232 1242 1242 1242 1242	A85A  A85C  A862  A868  A868  A87A  A889  A889  A889  A889  A899  A899  A899  A891  A899  A891  A890  A890	82 92 EA FZ EA DA 7A 6A 12 0A 12 22  8D 3F 34 10 BD 87 35 10 C1 08 22 48 5A 05 86 10 3D 0 8E6 84 2A 03 C4 70 21 5F 34 04 CA6 84 2B 01 4F 9A 86 AA E0 AA E0 AA 84 39	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET SET LA89C LA8AC	PULS  S IS A LOOI T 1 IS USEI FCB	A,PC  (-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$DA,\$CA,\$BA,\$AA,\$92 \$7A,\$6A,\$52,\$42,\$32,\$22 \$12,\$0A,\$0A,\$0B,\$02,\$02,\$0A \$12,\$22,\$32,\$42,\$52,\$6A  LABC1 X LB738 X #8 LABD5  LA895 #\$10  LA895 #\$70  SKP1  B LA90D ,X LA846  #\$0F GRBLOK ,S+ #\$80  GRBLOK ,S+ #\$80 ,X	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK  IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø)  \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF SAVE ONLY PIXEL ON/OFF INFO OR WITH WHICH PIXEL TO TURN ON OR IN THE COLOR FORCE GRAPHIC DISPLAY IT ON THE SCREEN
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1226 1227 1228 1230 1231 1232 1233 1234 1235 1236 1237 1238 1238 1239 1240 1241 1242	A85A  A85C  A862  A868  A87A  A888  A888  A888  A899  A891  A891  A891  A891  A894  A894  A894  A894  A895  A894  A896  A896  A897  A897  A897  A840	82 92 EA F2 EA DA 7A 6A 12 02  80 3F 34 10 BD 87 35 10 C1 08 22 48 63 05 86 10 30 08 22 48 24 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 21 5F 34 04 28 03 C4 70 30 68 84 28 01 4F 9A 86 AA 80 AA 80 AA 80 AA 84	FA CA 52 Ø2 32	FA BA 42 Ø2	FA F AA 9 32 2 Ø2 Ø	* (BI A LA85C 2 2 2 2 A A * SET SET	PULS  S IS A LOOI T 1 IS USEI FCB	A,PC  C-UP TABLE OF SINE VALUES D TO KEEP THE SERIAL OUTPU \$82,\$92,\$AA,\$BA,\$CA,\$DA \$EA,\$F2,\$FA,\$FA,\$FA,\$FA \$EA,\$A,\$CA,\$BA,\$AA,\$92 \$7A,\$CA,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB,\$CB	RESTORE OUTPUT CHARACTER AND RETURN  FOR THE TAPE DECK FSK IT MARKING)  6ET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK SAVE CHARACTER LOCATION SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB REGET CHARACTER LOCATION NIME ALLOWABLE COLORS ILLEGAL COLOR - ILLEGAL FUNCTION CALL CHANGE COLOR NUMBERS FROM Ø-8 TO (-1 TO 7) BRANCH IF SET (X,Y,Ø) \$10 OFFSET BETWEEN DIFFERENT COLORS MULT BY COLOR FOR TOTAL OFFSET GO SAVE THE COLOR GET CURRENT CHAR FROM SCREEN BRANCH IF NOT GRAPHIC SAVE ONLY THE COLOR INFO SKIP THE NEXT BYTE RESET ASCII BLOCK TO ZERO COLOR SAVE COLOR INFO SYNTAX CHECK FOR ) GET CURRENT CHARACTER FROM SCREEN BRANCH IF GRAPHIC RESET ASCII CHARACTER TO ALL PIXELS OFF SAVE ONLY PIXEL ON/OFF INFO OR WITH WHICH PIXEL TO TURN ON OR IN THE COLOR FORCE GRAPHIC

1247	A8B5	4 F			CLRA		* ACCA=ZERO GRAPHIC BLOCK - FOR USE IN CASE YOU RE
1248	AODS				CLIA		* TRYING TO RESET A NON GRAPHIC BLOCK
1249	A8B6	E6 8	4		LDB	,Х	GET CURRENT CHAR FROM SCREEN
1250	A8B8	2A F			BPL	LABAC	BRANCH IF NON-GRAPHIC
1251	A8BA	Ø3 8			COM	GRBLOK	INVERT PIXEL ON/OFF MASK
1252	A8BC	D4 8				GRBLOK	AND IT WITH CURRENT ON/OFF DATA
1253		E7 8			STB	, X	DISPLAY IT
1254	ABCØ	39			RTS	, .	
1255	71002	03					
1256				*** THIS	ROUTINE	WILL CHECK SYNTAX AND CH	ECK FOR LEGAL VALUES
1257						& POINT HORIZONTAL AND V	
1258						HEIR ABSOLUTE SCREEN ADDR	
1259						FOUR PIXELS OF THE GRAPH	
1260						IN GRBLOK.	
1261	A8C1	BD B	2 6A	LA8C1	JSR	LB26A	SYNTAX CHECK FOR "("
1262		BD Ø		LA8C4	JSR	RVEC21	HOOK INTO RAM
1263	A8C7	BD B			JSR	LB7ØB	EVALUATE EXPRESSION - RETURN VALUE IN ACCB
1264		C1 3			CMPB	#63	ONLY 64 HORIZONTAL GRAPHIC BLOCKS
1265		22 Ø			BHI	LA8D5	ILLEGAL FUNCTION CALL
1266		34 Ø				В	SAVE HOR COORD
1267		BD B			JSR	LB738	SYNTAX CHECK FOR COMMA AND EVALUATE EXPR
1268		C1 1			CMPB	#31	ONLY 32 VERTICAL BLOCKS
1269	A8D5	22 7		LA8D5	BHI	LA948	ILLEGAL FUNCTION CALL
1270	A8D7	34 Ø		LAODS	PSHS	В	SAVE VERT COORD
1271		54	7		LSRB	В	
1271	A8D9	J+		*	LJKD		DIVIDE BY TWO BECAUSE THERE ARE 2 GRAPHIC PIXELS/HOR CHARACTER POSITION (BYTE)
	APDA	96 2	ø		I DA	#32	
1273 1274	A8DA A8DC	86 2 3D	b .		LDA MUL	#32	32 BYTES/ROW GET ROW OFFSET OF CHAR POSITION
			1 00			#VIDDAM	
1275	A8DD AREØ	8E Ø			LDX	#VIDRAM	SCREEN BUFFER ADDRESS
1276	A8EØ	30 8			LEAX	D, X	ADD ROW OFFSET TO SCREEN BUFFER ADDRESS
1277	A8E2		±		LDB	1,5	GET HOR COORD
1278	A8E4	54			LSRB		2 VERTICAL PIXELS/CHARACTER POSITION ADD VERTICAL OFFSET TO CHARACTER ADDRESS
1279	A8E5	3A			ABX		
1280	A8E6	35 Ø			PULS	A,B	GET VER COORD TO ACCA, HOR COORD TO ACCB
1281	A8E8	84 Ø	1		ANDA	#1	KEEP ONLY LSB OF VER COORD
1282	A8EA	56			RORB		LSB OF HOR COORD TO CARRY FLAG
1283	A8EB	49			ROLA		LSB OF HOR TO BIT Ø OF ACCA
1284	A8EC	C6 1	Ø		LDB	#\$10	MAKE A BIT MASK - TURN ON BIT 4
1285	A8EE	54		LA8EE	LSRB		SHIFT IT RIGHT ONCE
1286	A8EF	4 A			DECA		SHIFTED IT ENOUGH?
1287	A8FØ	2A F			BPL	LA8EE	NO NO
1288	A8F2	D7 8	6		STB	GRBLOK	ACCB=8 FOR UPPER LEFT PIXEL, =4 FOR UPPER RIGHT
1289				*			PIXEL =2 FOR LOWER LEFT, =1 FOR LOWER RIGHT
1290	A8F4	39			RTS		
1291							
1292				* POINT			
1293	A8F5	8D C	D	POINT	BSR	LA8C4	EVALUATE EXPRESSION
1294	A8F7	C6 F			LDB	#\$FF	INITIAL VALUE OF ON/OFF FLAG = OFF (FALSE)
1295	A8F9	A6 8	4		LDA	, X	GET CURRENT GRAPHIC CHARACTER
1296	A8FB	2A Ø	D		BPL	LA9ØA	BRANCH IF NON-GRAPHIC (ALWAYS FALSE)
1297	A8FD	94 8	6		ANDA	GRBLOK	AND CURR CHAR WITH THE PIXEL IN QUESTION
1298	A8FF	27 Ø	8		BEQ	LA9Ø9	BRANCH IF THE ELEMENT IS OFF
1299	A9Ø1	E6 8	4		LDB	, Х	GET CURRENT CHARACTER
1300	A9Ø3	54			LSRB		* SHIFT RIGHT
1301	A9Ø4	54			LSRB		* SHIFT RIGHT
1302	A9Ø5	54			LSRB		* SHIFT RIGHT
1303	A9Ø6	54			LSRB		* SHIFT RIGHT - NOW THE HIGH NIBBLE IS IN THE LOW NIBBLE
1304	A9Ø7	C4 Ø	7		ANDB	#7	KEEP ONLY THE COLOR INFO
1305	A9Ø9	5C		LA9Ø9	INCB		ACCB=Ø FOR NO COLOR, =1 TØ 8 OTHERWISE
1306		BD A		LA9ØA	JSR	LA5E8	CONVERT ACCB TO FLOATING POINT
1307	A9ØD	7E B	2 67	LA9ØD	JMP	LB267	SYNTAX CHECK FOR )
1308							
1309							
				* CLS			
1310		BD Ø		* CLS CLS	JSR	RVEC22	HOOK INTO RAM
1310 1311	A913	27 1	3		JSR BEQ	RVEC22 LA928	BRANCH IF NO ARGUMENT
1310 1311 1312	A913 A915	27 1 BD B	3 7 ØB				
1310 1311 1312	A913	27 1 BD B	3 7 ØB		BEQ	LA928 LB7ØB	BRANCH IF NO ARGUMENT
1310 1311 1312 1313	A913 A915	27 1 BD B C1 Ø	3 7 ØB 8		BEQ JSR	LA928 LB7ØB	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB
1310 1311 1312 1313 1314	A913 A915 A918	27 1 BD B C1 Ø 22 1	3 7 ØB 8		BEQ JSR CMPB	LA928 LB7ØB #8	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >B, GO PRINT MICROSOFT SET FLAGS
1310 1311 1312 1313 1314 1315	A913 A915 A918 A91A	27 1 BD B C1 Ø 22 1 5D	3 7 ØB 8 B		BEQ JSR CMPB BHI	LA928 LB7ØB #8	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALIO ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT
1310 1311 1312 1313 1314 1315 1316	A913 A915 A918 A91A A91C	27 1 BD B C1 Ø 22 1 5D 27 Ø	3 7 ØB 8 B		BEQ JSR CMPB BHI TSTB	LA928 LB7ØB #8 LA937	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >B, GO PRINT MICROSOFT SET FLAGS
1310 1311 1312 1313 1314 1315 1316 1317	A913 A915 A918 A91A A91C A91D	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A	3 7 ØB 8 B		BEQ JSR CMPB BHI TSTB BEQ	LA928 LB7ØB #8 LA937	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø
1310 1311 1312 1313 1314 1315 1316 1317 1318	A913 A915 A918 A91A A91C A91D A91F	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1	3 7 ØB 8 B		BEQ JSR CMPB BHI TSTB BEQ DECB	LA928 LB7ØB #8 LA937	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7
1310 1311 1312 1313 1314 1315 1316 1317 1318	A913 A915 A918 A91A A91C A91D A91F A92Ø	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D	3 7 ØB 8 B		BEQ JSR CMPB BHI TSTB BEQ DECB LDA	LA928 LB7ØB #8 LA937	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >B, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319	A913 A915 A918 A91A A91C A91D A91F A92Ø A922	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø	3 7 ØB 8 B 6 Ø		BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL	LA928 LB70B #8 LA937 LA925	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320	A913 A915 A918 A91A A91C A91D A91F A92Ø A922 A923	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8	3 7 ØB 8 B 6 Ø	CLS	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB	LA928 LB70B #8 LA937 LA925 #\$10	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321	A913 A915 A918 A91A A91C A91D A91F A92Ø A922 A923 A925	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8	3 7 ØB 8 B 6 Ø	CLS	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322	A913 A915 A918 A91A A91C A91D A91F A92Ø A922 A923 A925	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8	3 7 ØB 8 B 6 Ø	CLS	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323	A913 A915 A918 A91A A91C A91D A91F A92Ø A922 A923 A925	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8	3 7 ØB 8 8 B 6 6 Ø	CLS LA925	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS
1310 1311 1312 1313 1314 1315 1316 1317 1318 1320 1321 1322 1323 1324 1325	A913 A915 A918 A91A A91C A91D A91F A920 A922 A923 A925 A927	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C	3 7 ØB 8 8 B 6 6 Ø F Ø	CLS  LA925  * CLEAR S	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80 SKP2	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326	A913 A915 A918 A91A A91C A91D A91F A920 A922 A923 A925 A927	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C	3 7 ØB 8 B 6 6 Ø F Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø	LA925 * CLEAR S	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB	LA928 LB708 #8 LA937 LA925 #\$10 #\$80 SKP2 #\$60 #VIDRAM	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327	A913 A915 A918 A91A A91C A91D A91F A92Ø A922 A923 A925 A927	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C C6 6 8E Ø 9F 8	3 7 ØB 8 8 6 6 Ø F Ø Ø 4 ØØ 8	LA925  * CLEAR S LA928  LA92D	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80 SKP2 #\$60 #VIDRAM CURPOS	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1327	A913 A915 A918 A910 A910 A917 A920 A922 A923 A925 A927	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C  C6 6 8E Ø 9F 8 E7 8	3 7 ØB 8 8 B 6 6 Ø F F Ø Ø Ø 4 4 Ø Ø 8 8 Ø Ø	LA925 * CLEAR S	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80 SKP2 #\$60 #VIDRAM CURPOS ,X+	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS FILL SCREEN WITH CONTENTS OF ACCB
1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1328	A913 A915 A918 A910 A910 A917 A920 A922 A923 A925 A927	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C  C6 6 8E Ø 9F 8 8C Ø	3 7 ØB 8 8 6 6 Ø F Ø Ø 4 4 ØØ 8 8 8 9 5 FF	LA925  * CLEAR S LA928  LA92D	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB FCB CREEN LDB LDX STX STX STB CMPX	LA928 LB708 #8 LA937 LA925 #\$10 #\$0F #\$80 SKP2 #\$60 #VIDRAM CURPOS ,X+ #VIDRAM+511	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS FILL SCREEN WITH CONTENTS OF ACCB END OF SCREEN?
1310 1311 1312 1313 1314 1315 1316 1317 1318 1320 1321 1322 1323 1324 1325 1326 1327 1328 1328 1328	A913 A915 A918 A910 A910 A917 A920 A922 A923 A925 A927 A928 A920 A928 A920 A921 A921 A921 A921 A921 A921 A921 A921	27 1 BD B C1 Ø 22 1 5D 5A 86 1 3D CA Ø CA 8 8C  C6 6 8E Ø 9F 8 E7 8 8C Ø 23 F	3 7 ØB 8 8 6 6 Ø F Ø Ø 4 4 ØØ 8 8 8 9 5 FF	LA925  * CLEAR S LA928  LA92D	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB CREEN LDX STX STB CMPX BLS	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80 SKP2 #\$60 #VIDRAM CURPOS ,X+	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS FILL SCREEN WITH CONTENTS OF ACCB
1310 1311 1312 1313 1314 1315 1316 1317 1318 1320 1321 1322 1323 1324 1325 1326 1327 1328 1327 1328 1329	A913 A915 A918 A910 A910 A917 A920 A923 A925 A927 A927 A928 A927 A928 A928 A928 A928 A928 A928 A928 A928	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C  C6 6 8E Ø 9F 8 E7 8 8C Ø 23 F	3 7 ØB 8 8 B 6 6 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø	LA925  * CLEAR S LA928 LA92B LA92D LA92F	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB CREEN LDX STX STX STX STX STX STX STX STX STX ST	LA928 LB70B #8 LA937 LA925 #\$10 #\$0F #\$80 SKP2 #\$60 #VIDRAM CURPOS ,X+ #VIDRAM+511 LA92F	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS FILL SCREEN WITH CONTENTS OF ACCB END OF SCREEN? NO
1310 1311 1312 1313 1314 1315 1316 1317 1320 1321 1322 1323 1324 1325 1326 1327 1328 1328 1329 1331 1331	A913 A915 A918 A910 A910 A917 A922 A923 A925 A927 A928 A927 A928 A929 A929 A931 A936 A937	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø 8C C6 6 8E Ø 9F 8 E7 8 8C Ø 23 F 39 8D E	3 7 ØB 8 8 8 6 6 Ø F Ø Ø 4 4 ØØ 8 8 Ø Ø 9 F F F F F F F F F F F F F F F F F	LA925  * CLEAR S LA928  LA92D	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB CREEN LDB STX STX STB CMPX BLS BSR	LA928 LB708 #8 LA937 LA925 #\$10 #\$0F #\$50 SKP2 #\$60 #VIDRAM CURPOS ,X+ #VIDRAM+511 LA92F	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS FILL SCREEN WITH CONTENTS OF ACCB END OF SCREEN?
1310 1311 1312 1313 1314 1315 1316 1317 1318 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333	A913 A915 A918 A910 A910 A917 A920 A922 A923 A925 A927 A927 A928 A920 A927 A931 A934 A936 A937 A939	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C  C6 6 8E Ø 9F 8 8C Ø 23 F 39 8D E 8E A	3 7 ØB 8 8 6 6 Ø F Ø Ø 4 4 ØØ 8 8 8 9 5 FF 9 9 F 1 65	LA925  * CLEAR S LA928 LA92B LA92D LA92F	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB FCB CREEN LDS STB LDX STX STX STB CMPX BLS RTS BSR LDX	LA928 LB708 #8 LA937 LA925 #\$10 #\$0F #\$80 SKP2 #\$60 #VIDRAM CURPOS ,X+ #VIDRAM+511 LA92F LA928 #LA166-1	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS FILL SCREEN WITH CONTENTS OF ACCB END OF SCREEN? NO  CLEAR SCREEN
1310 1311 1312 1313 1314 1315 1316 1317 1320 1321 1322 1323 1324 1325 1326 1327 1328 1328 1329 1331 1331	A913 A915 A918 A910 A910 A917 A922 A923 A925 A927 A928 A927 A928 A929 A929 A931 A936 A937	27 1 BD B C1 Ø 22 1 5D 27 Ø 5A 86 1 3D CA Ø CA 8 8C  C6 6 8E Ø 9F 8 8C Ø 23 F 39 8D E 8E A	3 7 ØB 8 8 6 6 Ø F Ø Ø 4 4 ØØ 8 8 8 9 5 FF 9 9 F 1 65	LA925  * CLEAR S LA928 LA92B LA92D LA92F	BEQ JSR CMPB BHI TSTB BEQ DECB LDA MUL ORB ORB FCB CREEN LDB STX STX STB CMPX BLS BSR	LA928 LB708 #8 LA937 LA925 #\$10 #\$0F #\$50 SKP2 #\$60 #VIDRAM CURPOS ,X+ #VIDRAM+511 LA92F	BRANCH IF NO ARGUMENT CALCULATE ARGUMENT, RETURN VALUE IN ACCB VALID ARGUMENT? IF ARGUMENT >8, GO PRINT MICROSOFT SET FLAGS COLOR Ø ACCB NOW CONTAINS Ø-7 EACH GRAPHIC BLOCK SEPARATED BY \$10 FROM ONE ANOTHER ACCB CONTAINS ONE OF 8 OFFSETS BITS Ø-3 SET FOR SOLID COLOR GRAPHIC BLOCK BIT 7 SET FOR GRAPHICS SKIP TWO BYTES  BLANK GET ADDR OF START OF SCREEN BUFFER SAVE IT IN CURPOS FILL SCREEN WITH CONTENTS OF ACCB END OF SCREEN? NO  CLEAR SCREEN

1336	A93F	BD B2	6D	ı	_A93F	JSR	LB26D	SYNTAX CHECK FOR A COMMA
1337	A942	BD B7			A942	JSR	LB7ØB	EVALUATE EXPRESSION, RETURN VALUE IN ACCB
1338	A945	5 D				TSTB		SET FLAGS
1339	A946	26 3C				BNE	LA984	RETURN IF NON ZERO
1340	A948	7E B4	4 A	L	A948	JMP	LB44A	ILLEGAL FUNCTION CALL IF ZERO
1341								
1342					SOUND			FULL HATE EVER FOR A CERTAIN (FREE HERE)
1343	A94B A94D	8D F5 D7 8C		S	SOUND	BSR STB	LA942 SNDTON	EVALUATE EXPRESSION (FREQUENCY)
1344 1345	A94F	8D EE				BSR	LA93F	SAVE IT EVALUATE EXPRESSION (SOUND LENGTH)
1346	A951	86 Ø4				LDA	#4	CONSTANT FACTOR
1347	A953	3D				MUL		EXPAND LENGTH EXPRESSION
1348	A954	DD 8D				STD	SNDDUR	SAVE LENGTH OF SOUND
1349	A956	B6 FF	Ø3			LDA	PIAØ+3	GET CONTROL REGISTER OF PIAØ, PORT B
1350	A959	8A Ø1				ORA	#1	*
1351	A95B	B7 FF				STA	PIAØ+3	* ENABLE 60 HZ INTERRUPT (PIA0 IRQ)
1352		ØF Ø8					ARYDIS	CLEAR THE ARRAY DISABLE FLAG - FOR NO APPARENT REASON
1353 1354	A96Ø A962	8D 4Ø 8D 12				BSR BSR	LA9A2 LA976	CONNECT D/A SOUND INPUT TO OUTPUT OF SOUND MUX TURN ON AUDIO - ENABLE SOUND MUX
1355	A964	8D 1F		1	_A964		LA985	STORE 2.5 VOLTS TO D/A AND WAIT
1356	A966	86 FE		-			#\$FE	DATA TO MAKE D/A OUT = 5 VOLTS
1357	A968	8D 1D					LA987	STORE IT TO D/A AND WAIT
1358	A96A	8D 19				BSR	LA985	STORE 2.5 VOLTS TO D/A AND WAIT
1359	A96C	86 Ø2				LDA	#2	DATA TO MAKE D/A OUT = Ø VOLTS
1360	A96E	8D 17				BSR	LA987	STORE IT TO D/A AND WAIT
1361	A97Ø	9E 8D				LDX	SNDDUR	* IS SNDDUR = 0? - THE IRQ INTERRUPT SERVICING
1362 1363	A972	26 FØ		•	•	BNE	LA964	* ROUTINE WILL DECREMENT SNDDUR NOT DONE YET
1364	A9/2	20 FØ				DNE	LA964	NOT DONE TEL
1365					THESE RO	UTINES	WILL ENABLE/DISABLE THE A	NALOG MUX
1366	A974	4 F				CLRA		BIT 3 OF ACCA = Ø, DISABLE ANALOG MUX
1367	A975	8C				FCB	SKP2	SKIP TWO BYTES
1368	A976	86 Ø8		L	A976	LDA	#8	BIT 3 OF ACCA = 1, ENABLE ANALOG MUX
1369	A978	A7 E2				STA	, - S	SAVE ACCA ON STACK
1370	A97A	B6 FF	23			LDA	PIA1+3	GET CONTROL REGISTER OF PIA1, PORT B
1371	A97D	84 F7				ANDA		RESET BIT 3
1372 1373	A97F A981	AA EØ B7 FF				ORA STA	,S+ PIA1+3	OR IN BIT 3 OF ACCA (SAVED ON STACK) SET/RESET CB2 OF U4
1374	A984	39	23	1	_A984	RTS	TIMITS	SETTRESET CDE OT 04
1375	A985	86 7E				LDA	#\$7E	DATA VALUE TO MAKE D/A OUTPUT = 2.5 VOLTS
1376	A987	B7 FF	20	L	A987	STA	DA	STORE IT IN D/A
1377	A98A	96 8C				LDA	SNDTON	GET FREQUENCY
1378	A98C	4 C		L	A98C	INCA		INCREMENT IT
1379	A98D	26 FD				BNE	LA98C	LOOP UNTIL DONE
1380	A98F	39				RTS		
1381 1382					* AUDIO			
1383	A99Ø	1F 89				TFR	A,B	SAVE ON/OFF TOKEN IN ACCB
1384	A992	9D 9F		,	10010	JSR	GETNCH	MOVE BASIC POINTER TO NEXT CHARACTER
1385	A994	C1 AA					#\$AA	OFF TOKEN?
1386	A996	27 DC				BEQ	LA974	YES - TURN OFF ANALOG MUX
1387		CØ 88				SUBB	#\$88	ON TOKEN
1388	A99A	BD A5	C9			JSR	LA5C9	SYNTAX ERROR IF NOT OFF OR ON
1389	A99D	5C				INCB	14040	NOW ACCB = 1
139Ø 1391	A99E A9AØ	8D Ø2 2Ø D4				BSR BRA	LA9A2 LA976	ROUTE CASSETTE TO SOUND MULTIPLEXER ENABLE SOUND MULTIPLEXER
1392	מאכא	20 04				DIA	LA370	ENABLE SOOND HOLITTEEXER
1393					* THIS ROU	JTINE W	ILL TRANSFER BIT Ø OF ACCB	TO SEL 1 OF
1394							TIPLEXER AND BIT 1 OF ACCB	
1395	A9A2	CE FF	01	L			#PIAØ+1	POINT U TO PIAØ CONTROL REG
1396	A9A5	8D ØØ				BSR	LA9A7	PROGRAM 1ST CONTROL REGISTER
	A9A7			L		LDA	, U	GET PIA CONTROL REGISTER
1398						ANDA	#\$F7	RESET CA2 (CB2) OUTPUT BIT
1399	A9AB A9AC					ASRB BCC	LAGRA	SHIFT ACCB BIT Ø TO CARRY FLAG BRANCH IF CARRY = ZERO
	A9AC A9AE					ORA		FORCE BIT 3=1; SET CA2(CB2)
	A9BØ			L		STA		PUT IT BACK IN THE PIA CONTROL REGISTER
1403	A9B2			-		RTS	,-	
1404								
1405				*	IRQ SERV	ICE		
	A9B3			Е	BIRQSV			CHECK FOR 60HZ INTERRUPT
	A9B6						LA9C5	RETURN IF 63.5 MICROSECOND INTERRUPT
1408	A9B8	BE ØØ					PIAØ+2 >SNDDUR	RESET PIAØ, PORT B INTERRUPT FLAG GET INTERRUPT TIMER (SOUND COMMAND)
	A9BE						LA9C5	RETURN IF TIMER = Ø
	A9CØ					LEAX		DECREMENT TIMER IF NOT = Ø
	A9C2						>SNDDUR	SAVE NEW TIMER VALUE
1413	A9C5			L		RTI		RETURN FROM INTERRUPT
1414								
1415					* JOYSTK			
	A9C6			J		JSR		EVALUATE JOYSTICK ARGUMENT
	A9C9			7 R		CMPB		TWO JOYSTICKS MAXIMUM (HOR & VER FOR EACH)
1418 1419			uЯ	/ D		LBHI TSTB	LUTTA	ILLEGAL FUNCTION CALL IF >3 SET FLAGS
	A9DØ						LA9D4	GET NEW DATA ONLY IF JOYSTK(Ø)
	A9D2						GETJOY	GET NEW DATA FOR ALL JOYSTICKS
	A9D4			L			#POTVAL	POINT X TO JOYSTICK DATA BUFFER
1423	4007	00 50				LDB	FPAØ+3	WHICH JOYSTICK DID YOU WANT?
1424		E6 85				LDB	В,Х	PUT ITS DATA INTO ACCB

```
1425
         A9DB 7F B4 F3
                                                   JMP
                                                          LB4F3
                                                                                          CONVERT ACCB INTO FLOATING POINT NUMBER
1426
                                       * JOYSTK DATA AT:
1427
                                                          $15B
                                                                                 $15D
1428
                                              $15A
                                                                      $15C
1429
                                              LEFT
                                                          LEFT
                                                                      RIGHT
                                                                                 RIGHT
                                                                                 HORIZ
                                                          HORIZ
                                                                      VERT
1430
                                              VERT
1431
1432
                                       ** THIS IS A 6 BIT SOFTWARE A/D CONVERSION ROUTINE
                                                                                          TURN OFF AUDIO
1433
         A9DF
                8D 94
                                       GETJOY
                                                   BSR
                                                           LA974
                                                                                          POINT X TO JOYSTICK DATA BUFFER
                8E Ø1 5E
                                                   LDX
                                                           #POTVAL+4
1434
         A9EØ
                C6 Ø3
                                                                                          GET FOUR SETS OF DATA (4 JOYSTICKS)
1435
         A9E3
                                                   LDB
1436
         A9E5
                86 ØA
                                       LA9E5
                                                   LDA
                                                           #1Ø
                                                                                          10 TRIES TO GET STABLE READING
                                                                                          STORE JOYSTICK NUMBER AND TRY NUMBER ON THE STACK
1437
         A9E7
                ED E3
                                                   STD
                                                           . - - S
         A9E9
                                                           LA9A2
                                                                                          SET THE SELECT INPUTS ON ANALOG MULTIPLEXER
1438
                8D B7
                                                   BSR
                CC 4Ø 8Ø
                                       LA9EB
                                                           #$4080
                                                                                          ACCA IS A SHIFT COUNTER OF HOW MANY BITS TO CONVERT
1439
         A9EB
                                                                                          AND WILL BE $40 (6 BITS) FOR THE COLOR COMPUTER. ACCB CONTAINS A VALUE EQUAL TO 1/2
1440
1441
1442
                                                                                          THE CURRENT TRIAL DIFFERENCE. INITIALLY =$80 (2.5 VOLTS).
1443
         A9EE
                A7 E2
                                       LA9EE
                                                   STA
                                                                                          TEMP STORE SHIFT COUNTER ON STACK
                                                           , - S
1444
         A9FØ
                CA Ø2
                                                   ORB
                                                           #2
                                                                                          KEEP RS 232 SERIAL OUT MARKING
                F7 FF 20
                                                                                          STORE IN D/A CONVERTER
         A9F2
1445
                                                   STB
                                                           DA
                                                                                          PUT R5232 OUTPUT BIT BACK TO ZERO
1446
         A9F5
                C8 Ø2
                                                   EORB
                                                           #2
                                                                                          HIGH BIT IS FROM COMPARATOR
BRANCH IF COMPARATOR OUTPUT IS HIGH
1447
         A9F7
                B6 FF ØØ
                                                   LDA
                                                           PTAØ
1448
         A9FA
                2B Ø3
                                                   BMI
                                                           LA9FF
                                                                                          SUBTRACT 1/2 THE CURRENT TRIAL DIFFERENCE
1449
         A9FC
                EØ E4
                                                   SUBB
1450
         A9FE
                                                           SKP2
                                                                                          SKIP NEXT TWO BYTES
                80
                                                   FCB
                                                                                          ADD 1/2 OF THE CURRENT TRIAL DIFFERENCE
PULL SHIFT COUNTER OFF THE STACK
1451
         A9FF
                EB E4
                                       LA9FF
                                                   ADDB
                                                           ,S
1452
         AAØ1
                A6 EØ
                                                   LDA
                                                           .S+
                                                                                          SHIFT IT RIGHT ONCE
1453
         AAØ3
                                                   LSRA
1454
         AA@4
                81 Ø1
                                                   СМРА
                                                          #1
                                                                                          HAVE ALL THE SHIFTS BEEN DONE?
1455
         AAØ6
                26 E6
                                                   BNE
                                                           LA9FF
                                                                                          NO
                                                                                          YES - THE DATA IS IN THE TOP 6 BYTES OF ACCB
1456
         AAØ8
                54
                                                   LSRB
         AAØ9
                                                   LSRB
                                                                                          PUT IT INTO THE BOTTOM SIX
                                                                                          IS THIS VALUE EQUAL TO THE LAST TRY?
YES - GO SAVE THE VALUE
1458
         AAØA
                E1 1F
                                                   СМРВ
                                                           -1,X
1459
         AAØC
                27 Ø4
                                                   BE0
                                                           LAA12
                                                                                          NO-DECREMENT TRIES COUNTER
1460
         AAØE
                6A E4
                                                   DEC
                                                           LA9EB
                                                                                          BRANCH IF YOU HAVEN T TRIED 10 TIMES
1461
         AA1Ø
                26 D9
                                                                                          IF YOU FALL THROUGH HERE YOU HAVE TRIED TO GET THE SAME READING 10 TIMES AND NEVER GOTTEN A MATCH. AS A RESULT YOU JUST FALL
1462
1463
                                                                                          THROUGH AND USE THE LAST VALUE READ IN.
1464
1465
         AA12
                E7 82
                                       LAA12
                                                   STB
                                                                                          SAVE THE DIGITIZED VALUE
1466
         AA14
                EC E1
                                                   LDD
                                                           , $++
                                                                                          GET THE NUMBER OF THE JOYSTICK JUST DONE
         AA16
                                                                                          DECR JOYSTK NUMBER
1467
                                                   DECB
                5A
1468
         AA17
                2A CC
                                                           LA9E5
                                                                                          BRANCH IF THE LAST ONE DONE WASN T NUMBER Ø
1469
         AA19
                39
                                                   RTS
1470
                                       * SET CARRY IF NUMERIC - RETURN WITH
1471
                                       * ZERO FLAG SET IF ACCA = Ø OR 3A(:) - END
1472
1473
                                       * OF BASIC LINE OR SUB LINE
                                                                                          IS THIS CHARACTER >=(ASCII 9)+1?
                81 3A
                                                          #'9+1
         AA1A
                                       BROMHK
                                                   CMPA
1474
1475
         AA1C
                24 ØA
                                                   BHS
                                                           LAA28
                                                                                          BRANCH IF > 9; Z SET IF = COLON
1476
         AA1E
                81 20
                                                   CMPA
                                                           #SPACE
                                                                                          SPACE?
                                                                                         NO - SET CARRY IF NUMERIC
IF SPACE, GET NECT CHAR (IGNORE SPACES)
1477
         AA2Ø
                26 02
                                                   RNF
                                                          I AA24
1478
         AA22
                ØE 9F
                                                          GETNCH
                                                   JMP
         AA24
                                                   SUBA
                                                                                          * SET CARRY IF
                                       LAA24
1480
         4426
                8Ø DØ
                                                   SUBA
                                                          #-'Ø
                                                                                          * CHARACTER > ASCII Ø
1481
         AA28
                39
                                       LAA28
                                                   RTS
1482
                                       * DISPATCH TABLE FOR SECONDARY FUNCTIONS
1483
1484
                                       * TOKENS ARE PRECEEDED BY $FF
                                                                                                 TOKEN #
                BC 7A
1485
         AA29
                                       LAA29
                                                   FDB
                                                          SGN
                                                                                          SGN
                                                                                                   80
1486
         AA2B
                BC EE
                                                   FDB
                                                           INT
                                                                                          INT
                                                                                                   81
1487
         AA2D
                BC 93
                                                   FDB
                                                           ABS
                                                                                          ABS
                                                                                                   82
1488
         AA2F
                Ø1 12
                                                   FDB
                                                           $0112
                                                                                          USR
                                                                                                   83
                BF 1F
1489
         AA31
                                                   FDB
                                                           RND
                                                                                          RND
                                                                                                   84
         AA33
                BF 78
1490
                                                   FDB
                                                           SIN
                                                                                          SIN
                                                                                                   85
1491
         AA35
                B7 5Ø
                                                   FDB
                                                           PEEK
                                                                                          PEEK
                                                                                                   86
1492
         AA37
                B6 81
                                                   FDB
                                                           LEN
                                                                                          LEN
                                                                                                   87
1493
         AA39
                B4 FD
                                                   FDB
                                                           STR
                                                                                          STR$
                                                                                                   88
1494
         AA3B
                B7 16
                                                           VAL
                                                                                                   89
1495
         AA3D
                B6 AØ
                                                   FDB
                                                           ASC
                                                                                          ASC
                                                                                                   88
1496
         AA3F
                B6 8C
                                                   FDB
                                                           CHR
                                                                                          CHR$
                                                                                                   88
1497
         AA41
                A5 CE
                                                   FDB
                                                                                          EOF
                                                           EOF
                                                                                                   80
1498
         AA43
                A9 C6
                                                           JOYSTK
                                                                                          JOYSTK
                                                                                                  8D
                                                   FDB
1499
         AA45
                B6 AB
                                                   FDB
                                                           LEFT
                                                                                          LEFT$
                                                                                                   8E
         AA47
                                                           RIGHT
                                                                                          RIGHT$
1500
                B6 C8
                                                   FDB
                                                                                                  8F
1501
         AA49
                B6 CF
                                                   FDB
                                                           MID
                                                                                          MID$
                                                                                                   90
1502
         AA4R
                A8 F5
                                                   FDB
                                                           POINT
                                                                                          POINT
                                                                                                   91
1503
         AA4D
                A5 64
                                                   FDB
                                                           INKEY
                                                                                          INKEYS
                                                                                                  92
         AA4F
                B4 EE
1504
                                                           MEM
                                                                                          MEM
                                                   FDB
1505
                                       * THIS TABLE CONTAINS PRECEDENCES AND DISPATCH ADDRESSES FOR ARITHMETIC
* AND LOGICAL OPERATORS - THE NEGATION OPERATORS DO NOT ACT ON TWO OPERANDS
* SØ THEY ARE NOT LISTED IN THIS TABLE. THEY ARE TREATED SEPARATELY IN THE
1506
1507
1508
1509
                                       * EXPRESSION EVALUATION ROUTINE. THEY ARE:
                                         UNARY NEGATION (-), PRECEDENCE $7D AND LOGICAL NEGATION (NOT), PRECEDENCE $5A
THE RELATIONAL OPERATORS <> = ARE ALSO NOT LISTED, PRECEDENCE $64.
1510
1511
                                       * A PRECEDENCE VALUE OF ZERO INDICATES END OF EXPRESSION OR PARENTHESES
1513
```

1514	AA51	79						LAA51	FCB	\$79	
1515	AA52	В9	C5					2717101	FDB	LB9C5	+
1516	AA54	79							FCB	\$79	
1517	AA55	В9	вс						FDB	LB9BC	-
1518		7 B							FCB	\$7B	
1519	AA58	ВА	CC						FDB	\$BACC	*
1520	AA5A	7B							FCB	\$7B	
1521	AA5B	ВВ	91						FDB	\$BB91	/
1522	AA5D	7 F							FCB	\$7F	
	AA5E	Ø1	1 D						FDB	\$Ø11D	EXPONENTIATION
	AA6Ø	50							FCB	\$50	
	AA61	B2	D5						FDB	\$B2D5	AND
	AA63	46							FCB	\$46	0.0
1527 1528	AA64	B2	υ4					*	FDB	LB2D4	OR
1529									THE RE	SERVED WORD TABLE	
1530								*	IIIL KL	SERVED WORD TABLE	TOKEN #
	AA66	46	4 F	n2				LAA66	FCC	'FO',\$8Ø+'R'	80
1532	AA69	47		-				2717100	FCC	'G',\$8Ø+'O'	81
	AA6B		45	CD					FCC	'RE',\$8Ø+'M'	82
1534	AA6E	Α7							FCB	'' + \$80	83
	AA6F		4 C	53	C5				FCC	'ELS',\$8Ø+'E'	84
1536	AA73	49	С6						FCC	'I',\$8Ø+'F'	85
1537	AA75	44	41	54	C1				FCC	'DAT',\$8Ø+'A'	86
1538	AA79	50	52	49	4 E	D4			FCC	'PRIN',\$8Ø+'T'	87
1539	AA7E	4 F	CE						FCC	'0',\$8Ø+'N'	88
1540	AA8Ø	49			55	D4			FCC	'INPU',\$8Ø+'T'	89
	AA85	45							FCC	'EN',\$8Ø+'D'	8A
	AA88	4 E			D4				FCC	'NEX',\$8Ø+'T'	8B
1543			49						FCC	'DI',\$8Ø+'M'	8C
	AA8F			41	C4				FCC	'REA',\$8Ø+'D'	8D
	AA93		55		- 4	4.5	- 0		FCC	'RU',\$8Ø+'N'	8E
	AA96		45	53	54	4 F	52		FCC	'RESTOR',\$80+'E'	8F
1547	AA9C	C5 52	4 5	E /		E 2	CE		FCC	'RETUR',\$8Ø+'N'	90
		53				32	CL		FCC	'STO',\$8Ø+'P'	91
	AAA7			4B					FCC	'POK',\$8Ø+'E'	92
		43							FCC	'CON',\$8Ø+'T'	93
	AAAF	40							FCC	'LIS',\$80+'T'	94
		43				D2			FCC	'CLEA',\$8Ø+'R'	95
	AAB8		45						FCC	'NE',\$8Ø+'W'	96
	AABB	43	4 C	4 F	41	C4			FCC	'CLOA',\$8Ø+'D'	97
1556	AACØ	43	53	41	56	C5			FCC	'CSAV',\$8Ø+'E'	98
1557	AAC5	4 F	50	45	CE				FCC	'OPE',\$8Ø+'N'	99
		43							FCC	'CLOS',\$8Ø+'E'	9A
	AACE				53	D4			FCC	'LLIS',\$8Ø+'T'	9B
	AAD3		45						FCC	'SE',\$8Ø+'T'	9C
	AAD6				45	D4			FCC	'RESE',\$8Ø+'T'	9D
1562 1563	AADB AADE		4C		4 F	D2			FCC FCC	'CL',\$8Ø+'S' 'MOTO',\$8Ø+'R'	9E 9F
	AAE3				4 F				FCC	'SOUN',\$8Ø+'D'	AØ
	AAE8				49				FCC	'AUDI',\$80+'0'	A1
	AAED			45		٠.			FCC	'EXE',\$8Ø+'C'	A2
1567	AAF1				50	С6			FCC	'SKIP',\$8Ø+'F'	A3
1568	AAF6			42					FCC	'TAB',\$8Ø+'('	A4
1569	AAFA	54	CF						FCC	'T',\$8Ø+'0'	A5
1570	AAFC	53	55	C2					FCC	'SU',\$8Ø+'B'	A6
1571	AAFF	54	48	45	CE				FCC	'THE',\$80+'N'	A7
1572	ABØ3		4F						FCC	'NO',\$8Ø+'T'	A8
1573	ABØ6			45	DØ				FCC	'STE',\$8Ø+'P'	A9
1574	ABØA		46	C6					FCC	'OF',\$8Ø+'F'	AA
1575	ABØD	AB							FCC	'+' + \$80	AB
1576 1577	ABØE ABØF	AD AA								'-' + \$80 '*' + \$80	AC AD
1578	AB1Ø	AF								'/' + \$80	AE
	AB11	DE								'^' + \$80	AF
			4 E	C.4						'AN',\$8Ø+'D'	ВØ
	AB15	4 F								'0',\$8Ø+'R'	B1
1582	AB17	ΒE								'>' + \$80	B2
1583	AB18	BD							FCB	'=' + \$80	B3
1584	AB19	BC							FCB	'<' + \$80	B4
1585								*			
1586								* TOKENS	FOR THE	SECONDARY FUNCTIONS ARE	
1587								*			TOKEN #
1588		53						LAB1A	FCC	'SG',\$8Ø+'N'	80
1589	AB1D								FCC	'IN',\$80+'T'	81
	AB2Ø								FCC	'AB',\$80+'S'	82 83
1591 1592	AB23 AB26	52	53 4F						FCC FCC	'US',\$8Ø+'R' 'RN',\$8Ø+'D'	84
1592	AB29									'SI',\$80+'N'	85
1593	AB20			45	CB				FCC	'PEE',\$8Ø+'K'	86
			45		20				FCC	'LE',\$80+'N'	87
1595	AB3Ø				Α4				FCC	'STR',\$8Ø+'\$'	88
	AB3Ø AB33	53	54	32						and the second s	
			41						FCC	'VA',\$8Ø+'L'	89
1596	AB33 AB37		41	CC						'AS',\$8Ø+'C'	89 8A
1596 1597	AB33 AB37 AB3A AB3D	56 41 43	41 53 48	CC C3 52						'AS',\$8Ø+'C' 'CHR',\$8Ø+'\$'	
1596 1597 1598	AB33 AB37 AB3A AB3D AB41	56 41 43 45	41 53 48 4F	CC C3 52 C6	Α4				FCC FCC FCC	'AS',\$8Ø+'C' 'CHR',\$8Ø+'\$' 'EO',\$8Ø+'F'	8A 8B 8C
1596 1597 1598 1599 1600 1601	AB33 AB37 AB3A AB3D AB41 AB44	56 41 43 45 4A	41 53 48 4F 4F	CC C3 52 C6 59	A4 53		СВ		FCC FCC FCC	'AS',\$80+'C' 'CHR',\$80+'\$' 'EO',\$80+'F' 'JOYST',\$80+'K'	8A 8B 8C 8D
1596 1597 1598 1599 1600	AB33 AB37 AB3A AB3D AB41	56 41 43 45 4A	41 53 48 4F 4F	CC C3 52 C6 59	A4 53		СВ		FCC FCC FCC	'AS',\$8Ø+'C' 'CHR',\$8Ø+'\$' 'EO',\$8Ø+'F'	8A 8B 8C

1603	AB4F	52	49	47	48	54	Α4		FCC	'RIGHT',\$80+'\$'	8F
1604	AB55			44					FCC	'MID',\$8Ø+'\$'	90
1605	AB59			49					FCC	'POIN',\$8Ø+'T'	91
1606	AB5E				45	59	Α4		FCC	'INKEY',\$8Ø+'\$'	92
1607	AB64	4 D	45	CD				*	FCC	'ME',\$8Ø+'M'	93
1608										FOR COMMANDS	TOVEN #
1609 1610	AB67	AD	47					* DISPAICE	FDB	FOR COMMANDS FOR	TOKEN # FOR 8Ø
1611	AB69	AE						LABO/	FDB	GO GO	GO 81
	AB6B	AE							FDB	REM	REM 82
1612	AB6D	AE							FDB	REM	REM 83
1613											
1614	AB6F	AE AF							FDB	REM IF	ELSE 84
1615	AB71								FDB		IF 85
1616	AB73	AE							FDB	DATA PRINT	DATA 86 PRINT 87
1617	AB75	B8 AF							FDB		
1618	AB77								FDB	ON	ON 88
1619	AB79	AF							FDB	INPUT	INPUT 89
1620	AB7B	AE							FDB	END	END 8A
1621 1622	AB7D	BØ							FDB	NEXT	NEXT 8B
1623	AB7F AB81	B3 BØ							FDB FDB	DIM READ	DIM 8C READ 8D
										RUN	
1624 1625	AB83 AB85	AE AD							FDB FDB	RESTOR	RUN 8E RESTORE 8F
1626	AB87	AE							FDB	RETURN	RETURN 90
1627	AB89	AE							FDB	STOP	STOP 91
1628	AB8B	B7							FDB	POKE	POKE 92
1629	AB8D	AE							FDB	CONT	CONTINUE93
1630	AB8F	B7							FDB	LIST	LIST 94
1631	AB91	AE							FDB	CLEAR	CLEAR 95
1632	AB93	AD							FDB	NEW	NEW 96
1633	AB95	A4							FDB	CLOAD	CLOAD 97
1634	AB97	A4							FDB	CSAVE	CSAVE 98
1635	AB99	A5							FDB	OPEN	OPEN 99
1636	AB9B	A4							FDB	CLOSE	CLOSE 9A
1637	AB9D	B7							FDB	LLIST	LLIST 9B
1638	AB9F	A8							FDB	SET	SET 9C
1639	ABA1	A8							FDB	RESET	RESET 9D
1640	ABA3	A9							FDB	CLS	CLS 9E
1641	ABA5	A7							FDB	MOTOR	MOTOR 9F
1642	ABA7	A9							FDB	SOUND	SOUND AØ
1643	ABA9	A9							FDB	AUDIO	AUDIO A1
1644	ABAB	A5							FDB	EXEC	EXEC A2
1645	ABAD	A5							FDB	SKIPF	SKIPF A3
1646	NUNU	N.S	LU					*	100	SKITT	SKITT AS
1647								* FRROR MI	FSSAGES	AND THEIR NUMBERS AS USED	INTERNALLY
1648	ABAF	4 E	46					LABAF	FCC	'NF'	Ø NEXT WITHOUT FOR
1649	ABB1	53						2712711	FCC	'SN'	1 SYNTAX ERROR
1650	ABB3	52							FCC	'RG'	2 RETURN WITHOUT GOSUB
1651	ABB5	4 F							FCC	'OD'	3 OUT OF DATA
1652	ABB7	46							FCC	'FC'	4 ILLEGAL FUNCTION CALL
1653	ABB9	4 F							FCC	'0V'	5 OVERFLOW
1654	ABBB	4 F							FCC	'OM'	6 OUT OF MEMORY
1655	ABBD	55							FCC	'UL'	7 UNDEFINED LINE NUMBER
1656	ABBF	42							FCC	'BS'	8 BAD SUBSCRIPT
1657	ABC1	44							FCC	'DD'	9 REDIMENSIONED ARRAY
1658	ABC3	2F							FCC	'/0'	10 DIVISION BY ZERO
1659	ABC5	49							FCC	'ID'	11 ILLEGAL DIRECT STATEMENT
1660	ABC7	54							FCC	'TM'	12 TYPE MISMATCH
1661	ABC9	4 F							FCC	'0S'	13 OUT OF STRING SPACE
1662	ABCB	4 C							FCC	'LS'	14 STRING TOO LONG
1663	ABCD	53	54						FCC	'ST'	15 STRING FORMULA TOO COMPLEX
1664	ABCF	43	4 E						FCC	'CN'	16 CAN'T CONTINUE
1665	ABD1								FCC	'FD'	17 BAD FILE DATA
1666	ABD3	41	4 F						FCC	'A0'	18 FILE ALREADY OPEN
1667	ABD5								FCC	'DN'	19 DEVICE NUMBER ERROR
1668	ABD7	49	4 F						FCC	'10'	20 I/O ERROR
1669	ABD9	46	4 D						FCC	'FM'	21 BAD FILE MODE
1670	ABDB	4 E	4 F						FCC	'NO'	22 FILE NOT OPEN
1671	ABDD	49	45						FCC	'IE'	23 INPUT PAST END OF FILE
1672	ABDF	44	53						FCC	'DS'	24 DIRECT STATEMENT IN FILE
1673											
1674	ABE1	20	45	52	52	4 F	52	LABE1	FCC	' ERROR'	
1675	ABE7	ØØ							FCB	\$00	
1676	ABE8		49	4 E	20			LABE8	FCC	' IN '	
1677	ABEC								FCB	\$00	
1678	ABED							LABED	FCB	CR	
1679	ABEE							LABEE	FCC	'OK'	
1680	ABFØ								FCB	CR,\$00	
1681	ABF2							LABF2	FCB	CR	
1682	ABF3		52	45	41	4B			FCC	'BREAK'	
1683	ABF8	ØØ							FCB	\$00	
1684										CK FOR GOSUB/RETURN OR I	
1685										INDEX VARIABLE DESCRIPTOR	
1686										ED IN VARDES. EACH BLOCK OF	
1687										80 LEADER BYTE AND THE GOS	
1688										ADER BYTE. THE FIRST NON '	"FOR/NEXT" DATA
1689		_								GOSUB/RETURN	
1690	ABF9	30	64					LABF9	LEAX	4,S	POINT X TO 3RD ADDRESS ON STACK - IGNORE THE
1691								*			FIRST TWO RETURN ADDRESSES ON THE STACK

1692	ABFB				LABFB	LDB	#18	18 BYTES SAVED ON STACK FOR EACH FOR LOOP
1693	ABFD					STX	TEMPTR	SAVE POINTER
1694	ABFF					LDA	, X	GET 1ST BYTE
1695	ACØ1					SUBA	#\$80	* CHECK FOR TYPE OF STACK JUMP FOUND
1696	ACØ3					BNE	LAC1A	* BRANCH IF NOT FOR/NEXT
1697	ACØ5					LDX	1,X	= GET INDEX VARIABLE DESCRIPTOR
1698	ACØ7					STX	TMPTR1	= POINTER AND SAVE IT IN TMPTR1
1699	ACØ9					LDX	VARDES	GET INDEX VARIABLE BEING SEARCHED FOR
1700	ACØB	27	Ø9			BEQ	LAC16	BRANCH IF DEFAULT INDEX VARIABLE - USE THE
1701					*			FIRST FOR/NEXT DATA FOUND ON STACK
1702					*			IF NO INDEX VARIABLE AFTER NEXT
1703	ACØD	9 C	11			CMPX	TMPTR1	DOES THE STACK INDEX MATCH THE ONE
1704					*			BEING SEARCHED FOR?
1705	ACØF	27	<b>0</b> 9			BEQ	LAC1A	YES
1706	AC11	9 E	Ø F			LDX	TEMPTR	* RESTORE INITIAL POINTER, ADD
1707	AC13	3A				ABX		* 18 TO IT AND LOOK FOR
1708	AC14	20	E5			BRA	LABFB	* NEXT BLOCK OF DATA
1709	AC16	9 E	11		LAC16	LDX	TMPTR1	= GET 1ST INDEX VARIABLE FOUND AND
1710	AC18	9 F	3B			STX	VARDES	= SAVE AS NEXT INDEX
1711	AC1A	9 E	Ø F		LAC1A	LDX	TEMPTR	POINT X TO START OF FOR/NEXT DATA
1712	AC1C	4 D				TSTA		SET ZERO FLAG IF FOR/NEXT DATA
1713	AC1D	39				RTS		
1714					* CHECK	FOR MEMO	ORY SPACE FOR NEW TOP OF	
1715							E ARRAYS TO NEW LOCATIO	
1716	AC1E	8D	17		LAC1E	BSR	LAC37	ACCD = NEW BOTTOM OF FREE RAM - IS THERE
1717					*			ROOM FOR THE STACK?
1718					* MOVE B	YTES FRO	OM V43(X) TO V41(U) UNTI	
1719							UE OF U IN V45	
1720	AC2Ø	DF ·	41		LAC2Ø	LDU	V41	POINT U TO DESTINATION ADDRESS (V41)
1721	AC22					LEAU		ADD ONE TO U - COMPENSATE FOR FIRST PSHU
1722	AC24					LDX	V43	POINT X TO SOURCE ADDRESS (V43)
	AC26					LEAX		ADD ONE - COMPENSATE FOR FIRST LDA ,X
	AC28				LAC28	LDA	,-X	GRAB A BYTE FROM SOURCE
1725	AC2A				LAGEO		A	MOVE IT TO DESTINATION
1726	AC2C					CMPX		DONE?
1727	AC2E					BNE	LAC28	NO - KEEP MOVING BYTES
1728	AC3Ø					STU	V45	SAVE FINAL DESTINATION ADDRESS
1729	AC32		+5		LAC32	RTS	V45	SAVE FINAL DESTINATION ADDRESS
1729	MUSZ	33					IF THERE IS ROOM TO STOR	E 2+ACCB
								E 2"ACCB
1731	4022	4 =					RAM - OM ERROR IF NOT	+ ACCD CONTAINS NUMBER OF EVERA
1732	AC33				LAC33	CLRA		* ACCD CONTAINS NUMBER OF EXTRA
1733	AC34					ASLB	ADVEND	* BYTES TO PUT ON STACK
1734	AC35						ARYEND	END OF PROGRAM AND VARIABLES
1735	AC37			A	LAC37	ADDD		ADD STACK BUFFER - ROOM FOR STACK?
1736	AC3A			_		BCS	LAC44	BRANCH IF GREATER THAN \$FFFF
1737	AC3C					STS	BOTSTK	CURRENT NEW BOTTOM OF STACK STACK POINTER
1738	AC3F			7			BOTSTK	ARE WE GOING TO BE BELOW STACK?
1739	AC42					BCS	LAC32	YES - NO ERROR
1740	AC44	Cb :	øC.		LAC44	LDB	#6*2	OUT OF MEMORY ERROR
1741					* FDDOD	CEDVICI	IC DOUTING	
1742	4046		~ -	-			NG ROUTINE	HOOK THIS DAM
1743 1744	AC46 AC49				LAC46	JSR	RVEC16	HOOK INTO RAM
1/44		BU !				JSR	RVEC17	HOOK INTO RAM
1745		D D		9		JSR	LA7E9 LA974	TURN OFF CASSETTE DISABLE ANA MUX
1745	AC4C			4				
1746	AC4C AC4F	BD .	49 7			JSR		
1746 1747	AC4C AC4F AC52	BD .	A9 7			JSR	LAD33	RESET STACK, STRING STACK, CONTINUE POINTER
1746 1747 1748	AC4C AC4F AC52 AC55	BD . BD . ØF	A9 7 AD 3 6F	3		JSR CLR	LAD33 DEVNUM	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN
1746 1747 1748 1749	AC4C AC4F AC52 AC55 AC57	BD BD BD BD BD	A9 7 AD 3 6F B9 5	3 C		JSR CLR JSR	LAD33 DEVNUM LB95C	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN
1746 1747 1748 1749 1750	AC4C AC4F AC52 AC55 AC57 AC5A	BD BD BD BD	A9 7 AD 3 6F B9 5	3 C F		JSR CLR JSR JSR	LAD33 DEVNUM LB95C LB9AF	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN
1746 1747 1748 1749 1750 1751	AC4C AC4F AC52 AC55 AC57 AC5A AC5D	BD BD BD BD BE	A9 7 AD 3 6F B9 5	3 C F		JSR CLR JSR JSR LDX	LAD33 DEVNUM LB95C	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE
1746 1747 1748 1749 1750 1751 1752	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC6Ø	BD BD BD BD BE 3A	A9 7 AD 3 6F B9 5 B9 A	3 C F		JSR CLR JSR JSR LDX ABX	LAD33 DEVNUM LB95C LB9AF #LABAF	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET
1746 1747 1748 1749 1750 1751 1752 1753	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC6Ø AC61	BD BD BD BD BD BE 3A BD	A9 7 AD 3 6F B9 5 B9 A AB A	3 C F		JSR CLR JSR JSR LDX ABX BSR	LAD33 DEVNUM LB95C LB9AF #LABAF	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A ? TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND
1746 1747 1748 1749 1750 1751 1752 1753 1754	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC6Ø AC61 AC63	BD ØF BD BD BD 8E 3A 8D 8D	A9 7 AD 3 6F B9 5 B9 4 AB 4	3 C F F		JSR CLR JSR JSR LDX ABX BSR BSR	LAD33 DEVNUM LB95C LB9AF #LABAF LACAØ LACAØ	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN)
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC6Ø AC61 AC63 AC65	BD ØF BD BD SE 3A 8D 8E 8E	A9 7 AD 3 6F B9 5 B9 A AB A 3D 3B	3 C F F		JSR CLR JSR JSR LDX ABX BSR BSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LACAØ	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC6Ø AC61 AC63 AC65 AC68	BD ØF BD BD SE SA SD SE BD	A9 7 AD 3 6F B9 5 B9 4 AB 4 3D 3B AB 6	3 C F F	LAC68	JSR CLR JSR JSR LDX ABX BSR BSR LDX JSR	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC6Ø AC61 AC63 AC65 AC68 AC68	BD B	A9 7 AD 3 6F B9 5 B9 4 AB 4 3D 3B AB 6	3 C F F	LAC68	JSR CLR JSR JSR LDX ABX BSR BSR LDX JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LACAØ	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN)
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC60 AC61 AC63 AC65 AC68 AC68 AC6B	BD BD BD BD BD BE 3A BD	A9 7 AD 3 6F B9 5 B9 4 AB 4 3D 3B B9 9	3 C F F	LAC68	JSR CLR JSR JSR LDX ABX BSR BSR LDX JSR LDX JSR LDA INCA	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC63 AC65 AC68 AC6B AC6B AC6B	BD	A9 7 AD 3 66F BB9 5 AAB A AB B9 9	3 C F F F	LAC68	JSR CLR JSR JSR LDX ABX BSR BSR LDX JSR LDX JSR LDA INCA BEQ	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC60 AC61 AC63 AC65 AC68 AC68 AC6B	BD	A9 7 AD 3 66F BB9 5 AAB A AB B9 9	3 C F F F	LAC68	JSR CLR JSR JSR LDX ABX BSR BSR LDX JSR LDX JSR LDA INCA	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC63 AC65 AC68 AC6B AC6B AC6B	BD	A9 7 AD 3 66F BB9 5 AAB A AB B9 9	3 C F F F		JSR CLR JSR JSR LDX ABX BSR BSR LDX JSR LDA LDA INCA BEQ JSR	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1761	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC60 AC61 AC63 AC65 AC68 AC68 AC6B AC6B AC6D	BD	A9 7 7 7 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9	3 C F F Ø C	* THIS I	JSR CLR JSR JSR LDX BSR BSR LDX JSR LDA INCA BEQ JSR S THE MA	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****
1746 1747 1748 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC63 AC65 AC68 AC6B AC6B AC6E AC70	BD	A9 7 3 6 6 F F F F F F F F F F F F F F F F F	3 C F F Ø C C		JSR CLR JSR JSR LDSR LDX JSR LDX JSR LDA INCA BEQ JSR S THE M/ JSR	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5  AIN LOOP OF BASIC WHEN I LB95C	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A 0R TO SCREEN SEND A 7 TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC61 AC63 AC65 AC68 AC68 AC6B AC6B AC6D AC6C AC7Ø	BD	A9 7 7 AAD 3 6 F F F F F F F F F F F F F F F F F F	3 CFFFF ØCC	* THIS I	JSR CLR JSR JSR LDX ABX BSR LDX JSR LDA INCA BEQ JSR STHE M/ JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN **** N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1762 1763	AC4C AC4F AC52 AC55 AC57 AC5A AC5D AC60 AC61 AC63 AC65 AC68 AC6B AC6B AC6B AC6B AC6C AC70	BD	A9 7 7 8 8 8 9 9 9 9 9 8 8 8 9 9 9 9 9 9	3 C F F Ø C C	* THIS I LAC73	JSR CLR JSR LDX ABX BSR LDX JSR LDX JSR LDA INCA BEQ JSR S THE M/ JSR LDX JSR LDX JSR	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1765 1765	AC4C AC4F AC55 AC57 AC5A AC5D AC61 AC63 AC65 AC68 AC6B AC6B AC6E AC7Ø	BD	A9 7 7 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9	3 C F F C C D C C G G G G G G G G G G G G G G G	* THIS I	JSR CLR JSR JSR LDX ABX BSR LDX JSR LDA INCA BEQ JSR STHE M/ JSR LDX JSR LDA JSR LDA INCA JSR LDA INCA JSR LDA INCA JSR LDA INCA JSR LDA JSR LDA JSR LDA JSR LDA	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5  AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C #LABE1-1 LB99C LA330	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A QR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK, CR MESSAGE PRINT OK, CR GO GET AN INPUT LINE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1763 1764 1765 1766 1767	AC4C AC4F AC55 AC57 AC5A AC5A AC60 AC61 AC65 AC68 AC6B AC6B AC6B AC6B AC6B AC6C AC70	BD  ØF  ØF  BD  BB  BC  BC	A9 7 7 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9	3 C F F C C D C C G G G G G G G G G G G G G G G	* THIS I LAC73	JSR CLR JSR LDX ABX BSR LDX JSR LDX JSR LDX JSR LDA INCA BEQ JSR STHE MA JSR LDX JSR LDU	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA39Ø #FFFF	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1765 1766 1765 1766	AC4C AC4F AC55 AC57 AC5A AC5D AC60 AC61 AC63 AC65 AC6B AC6B AC6B AC6D AC6C AC70 AC70	BD	49 7 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	3 C F F C C D C C G G G G G G G G G G G G G G G	* THIS I LAC73	JSR CLR JSR JSR LDX ABX BSR LDX JSR LDA INCA BEQ JSR STHE M/ JSR LDX JSR LDU STU	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA330 #FFFF CURLIN	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN
1746 1747 1748 1759 1759 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1767 1768 1766 1767	AC4C AC4F AC55 AC57 AC57 AC5A AC60 AC61 AC63 AC65 AC68 AC6B AC6B AC6B AC70 AC73 AC73 AC73 AC76 AC77 AC76 AC77	BD  ØF  BD  BB  BB	49 7 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	3 C F F C C D C C G G G G G G G G G G G G G G G	* THIS I LAC73	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR STHE M/ JSR LDX JSR LDX STU BCS	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB95C #LABE1-1 LB99C LA39Ø #FFFF CURLIN LAC7C	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A QR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN BRANCH IF LINE INPUT TERMINATED BY BREAK
1746 1747 1748 1759 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1766 1767 1768 1767	AC4C AC4F AC52 AC55 AC57 AC5A AC5A AC5A AC5A AC6A AC6B AC6B AC6B AC6B AC6B AC78 AC78 AC78 AC78 AC74 AC74 AC74 AC74 AC74 AC74 AC74 AC74	BD	A9 7 7 8 8 8 9 9 9 9 8 8 9 9 9 9 9 9 9 9	3 CFFF F ØCC D CC D CC ØFF	* THIS I LAC73	JSR CLR JSR LDX ABX BSR BSR LDA INCA INCA BEQ JSR LDX JSR LDU STU BCS TST	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5  AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA39Ø #FFFF CURLIN LAC7C CINBFL	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A QR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO COMSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK, CR MESSAGE PRINT OK, CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN BRANCH IF LINE INPUT TERMINATED BY BREAK CHECK CONSOLE INPUT BUFFER STATUS
1746 1747 1748 1759 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1767 1768 1769 1770	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC61 AC62 AC68 AC67 AC73 AC73 AC74 AC82 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC88 AC88 AC88 AC88 AC88 AC88 AC88	BD	A9 7 7 8 8 8 9 9 9 8 8 9 9 9 9 9 9 9 9 9	3 CFFF F ØCC D CC D CC ØFF	* THIS I LAC73	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LDX JSR LDA INCA BEQ JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA39Ø #FFFF CURLIN LAC7C CINBFL LA4BF	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN BRANCH IF LINE INPUT TERMINATED BY BREAK CHECK CONSOLE INPUT BUFFER STATUS BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY
1746 1747 1748 1759 1750 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1765 1766 1767 1768 1768 1769 1768 1769 1768 1769 1771 1772	AC4C AC4F AC52 AC55 AC57 AC5A AC5A AC5A AC5A AC6A AC6B AC6B AC6B AC6B AC6B AC6B AC78 AC78 AC78 AC78 AC74 AC74 AC74 AC74 AC74 AC74 AC74 AC74	BD	A9 7 7 8 8 8 9 9 9 8 8 9 9 9 9 9 9 9 9 9	3 CFFF F ØCC D CC D CC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDA INCA INCA BEQ JSR LDX JSR LDU STU BCS TST	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5  AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA39Ø #FFFF CURLIN LAC7C CINBFL	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A QR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK, CR MESSAGE PRINT OK, CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFF SAVE IT IN CURLIN BRANCH IF LINE INPUT TERMINATED BY BREAK CHECK CONSOLE INPUT BUFFER STATUS BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY SAVE (X) AS CURRENT INPUT POINTER - THIS WILL
1746 1747 1748 1759 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1767 1768 1769 1770	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC61 AC62 AC68 AC67 AC73 AC73 AC74 AC82 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC88 AC88 AC88 AC88 AC88 AC88 AC88	BD	A9 7 7 8 8 8 9 9 9 8 8 9 9 9 9 9 9 9 9 9	3 CFFF F ØCC D CC D CC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LDX JSR LDA INCA BEQ JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA39Ø #FFFF CURLIN LAC7C CINBFL LA4BF	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN BRANCH IF LINE INPUT TERMINATED BY BREAK CHECK CONSOLE INPUT BUFFER STATUS BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY
1746 1747 1748 1759 1750 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1765 1766 1767 1768 1768 1769 1768 1769 1768 1769 1771 1772	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC61 AC62 AC68 AC67 AC73 AC73 AC74 AC82 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC88 AC88 AC88 AC88 AC88 AC88 AC88	BD	A9 7 7 8 8 8 9 9 9 8 8 9 9 9 9 9 9 9 9 9	3 CFFF F ØCC D CC D CC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LDX JSR LDA INCA BEQ JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA39Ø #FFFF CURLIN LAC7C CINBFL LA4BF	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A QR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK, CR MESSAGE PRINT OK, CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFF SAVE IT IN CURLIN BRANCH IF LINE INPUT TERMINATED BY BREAK CHECK CONSOLE INPUT BUFFER STATUS BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY SAVE (X) AS CURRENT INPUT POINTER - THIS WILL
1746 1747 1748 1759 1759 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1766 1767 1768 1767 1768 1769 1769 1770 1770 1770	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC61 AC62 AC68 AC67 AC73 AC73 AC74 AC82 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC86 AC84 AC88 AC88 AC88 AC88 AC88 AC88 AC88	BD	A9 7 3 4 5 6 6 6 6 7 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6	3 CFFF F ØCC D CC D CC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LDX JSR LDA INCA BEQ JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA39Ø #FFFF CURLIN LAC7C CINBFL LA4BF	RESET STACK, STRING STACK, CONTINUE POINTER  SET DEVICE NUMBER TO SCREEN  SEND A QR TO SCREEN  SEND A ? TO SCREEN  POINT TO ERROR TABLE  ADD MESSAGE NUMBER OFFSET  * GET TWO CHARACTERS FROM X AND  * SEND TO CONSOLE OUT (SCREEN)  POINT TO "ERROR" MESSAGE  PRINT MESSAGE POINTED TO BY X  GET CURRENT LINE NUMBER (CURL IN)  TEST FOR DIRECT MODE  BRANCH IF DIRECT MODE  PRINT IN ****  N DIRECT MODE  MOVE CURSOR TO START OF LINE  POINT X TO OK, CR MESSAGE  PRINT OK, CR  GO GET AN INPUT LINE  THE LINE NUMBER FOR DIRECT MODE IS \$FFFF  SAVE IT IN CURLIN  BRANCH IF LINE INPUT TERMINATED BY BREAK  CHECK CONSOLE INPUT BUFFER STATUS  BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY  SAVE (X) AS CURRENT INPUT POINTER - THIS WILL  ENABLE THE LIVE KEYBOARD (DIRECT) MODE. THE
1746 1747 1748 1759 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1770 1770 1771 1772 1773	AC4C AC4F AC52 AC55 AC57 AC5A AC60 AC61 AC61 AC61 AC62 AC66 AC79 AC77 AC77 AC77 AC77 AC77 AC77 AC77	BD BD BBD	A9 7 3 3 4 A B B B B B B B B B B B B B B B B B B	3 CFFF F ØCC D CC D CC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX INCA INCA BEQ JSR LDX JSR LDX JSR LDA INCA BEQ JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5  AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA390 #FFFF CURLIN LAC7C CINBFL LA4BF CHARAD	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN BRANCH IF LINE INPUT TERMINATED BY BREAK CHECK CONSOLE INPUT BUFFER STATUS BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY SAVE (X) AS CURRENT INPUT POINTER - THIS WILL ENABLE THE LIVE KEYBOARD CORRECT MODE. THE LINE JUST ENTERED WILL BE INTERPRETED
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1770 1771 1772 1773 1774 1777	AC4C AC4F AC55 AC55 AC57 AC5A AC60 AC61 AC63 AC68 AC68 AC60 AC67 AC70 AC70 AC70 AC70 AC70 AC70 AC70 AC7	BD	A9 7 3 3 4 5 5 6 5 6 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6	3 CFFF  ØCC  CDCC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LDA INCA BEQ JSR TST LDX JSR LDX JSR LDA INCA BEQ JSR TST LDX JSR JSR JSR LDX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA33Ø #FFFF CURLIN LAC7C CINBFL LA4BF CHARAD	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A CR TO SCREEN SEND A ? TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO CONSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK , CR MESSAGE PRINT OK , CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY SAVE (X) AS CURRENT INPUT POINTER - THIS WILL ENABLE THE LIVE KEYBOARD (DIRECT) MODE. THE LINE JUST ENTERED WILL BE INTERPRETED GET NEXT CHARACTER FROM BASIC
1746 1747 1748 1759 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1770 1770 1770 1770 1771 1772 1773 1774 1775	AC4C AC4F AC55 AC57 AC5A AC60 AC61 AC63 AC68 AC68 AC68 AC79 AC77 AC72 AC74 AC74 AC84 AC86 AC88 AC88 AC88 AC88	BD	A9 7 3 3 4 5 5 6 5 6 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6	3 CFFF  ØCC  CDCC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LBNE STX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA390 #FFFF CURLIN LAC7C CINBFL LA4BF CHARAD	RESET STACK, STRING STACK, CONTINUE POINTER  SET DEVICE NUMBER TO SCREEN  SEND A QR TO SCREEN  SEND A ? TO SCREEN  POINT TO ERROR TABLE  ADD MESSAGE NUMBER OFFSET  * GET TWO CHARACTERS FROM X AND  * SEND TO CONSOLE OUT (SCREEN)  POINT TO "ERROR" MESSAGE  PRINT MESSAGE POINTED TO BY X  GET CURRENT LINE NUMBER (CURL IN)  TEST FOR DIRECT MODE  BRANCH IF DIRECT MODE  PRINT IN ****  N DIRECT MODE  MOVE CURSOR TO START OF LINE  POINT X TO OK, CR MESSAGE  PRINT OK, CR  GO GET AN INPUT LINE  THE LINE NUMBER FOR DIRECT MODE IS \$FFFF  SAVE IT IN CURLIN  BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY  SAVE (X) AS CURRENT INPUT POINTER - THIS WILL  ENABLE THE LIVE KEYBOARD (DIRECT) MODE. THE  LINE JUST ENTERED WILL BE INTERPRETED  GET NEXT CHARACTER FROM BASIC  NO LINE INPUT - GET ANOTHER LINE
1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1760 1761 1762 1763 1764 1765 1766 1767 1768 1767 1778 1770 1771 1772 1773 1774 1775 1776 1777	AC4C AC4F AC55 AC57 AC5A AC60 AC61 AC63 AC68 AC68 AC68 AC79 AC77 AC72 AC74 AC74 AC84 AC86 AC88 AC88 AC88 AC88	BD	A9 7 3 3 4 5 5 6 5 6 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6	3 CFFF  ØCC  CDCC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LBNE STX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA390 #FFFF CURLIN LAC7C CINBFL LA4BF CHARAD	RESET STACK, STRING STACK, CONTINUE POINTER SET DEVICE NUMBER TO SCREEN SEND A ? TO SCREEN SEND A ? TO SCREEN POINT TO ERROR TABLE ADD MESSAGE NUMBER OFFSET * GET TWO CHARACTERS FROM X AND * SEND TO COMSOLE OUT (SCREEN) POINT TO "ERROR" MESSAGE PRINT MESSAGE POINTED TO BY X GET CURRENT LINE NUMBER (CURL IN) TEST FOR DIRECT MODE BRANCH IF DIRECT MODE PRINT IN ****  N DIRECT MODE MOVE CURSOR TO START OF LINE POINT X TO OK, CR MESSAGE PRINT OK, CR GO GET AN INPUT LINE THE LINE NUMBER FOR DIRECT MODE IS \$FFFF SAVE IT IN CURLIN BRANCH IF LURE INPUT TERMINATED BY BREAK CHECK CONSOLE INPUT BUFFER STATUS BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY SAVE (X) AS CURRENT INPUT POINTER - THIS WILL ENABLE THE LIVE KEYBOARD (DIRECT) MODE. THE LINE JUST ENTERED WILL BE INTERPRETED GET NEXT CHARACTER FROM BASIC NO LITE INPUT FOR HARACTER FROM BASIC NO LITE INPUT INPUT FOR LINE BRANCH IF NUMERIC - THERE WAS A LINE NUMBER BEFORE
1746 1747 1748 1759 1750 1751 1752 1753 1754 1755 1756 1757 1758 1769 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778	AC4C AC4F AC55 AC57 AC5A AC60 AC61 AC63 AC68 AC68 AC68 AC79 AC77 AC72 AC74 AC74 AC84 AC86 AC88 AC88 AC88 AC88	BD	A9 73 3 4 5 5 6 6 6 7 6 6 6 7 6 6 6 7 6 6 6 6 7 6 6 6 6 7 6 6 6 6 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	3 CFFF  ØCC  CDCC ØFF	* THIS I LAC73 LAC7C	JSR CLR JSR LDX ABX BSR BSR LDX JSR LDA INCA BEQ JSR LDX JSR LBNE STX	LAD33 DEVNUM LB95C LB9AF #LABAF  LACAØ LACAØ #LABE1-1 LB99C CURLIN  LAC73 LBDC5 AIN LOOP OF BASIC WHEN I LB95C #LABE1-1 LB99C LA390 #FFFF CURLIN LAC7C CINBFL LA4BF CHARAD	RESET STACK, STRING STACK, CONTINUE POINTER  SET DEVICE NUMBER TO SCREEN  SEND A CR TO SCREEN  SEND A ? TO SCREEN  SEND A ? TO SCREEN  POINT TO ERROR TABLE  ADD MESSAGE NUMBER OFFSET  * GET TWO CHARACTERS FROM X AND  * SEND TO CONSOLE OUT (SCREEN)  POINT TO "ERROR" MESSAGE  PRINT MESSAGE POINTED TO BY X  GET CURRENT LINE NUMBER (CURL IN)  TEST FOR DIRECT MODE  BRANCH IF DIRECT MODE  PRINT IN ****  N DIRECT MODE  MOVE CURSOR TO START OF LINE  POINT X TO OK , CR MESSAGE  PRINT OK , CR  GO GET AN INPUT LINE  THE LINE NUMBER FOR DIRECT MODE IS \$FFFF  SAVE IT IN CURLIN  BRANCH IF LINE INPUT TERMINATED BY BREAK  CHECK CONSOLE INPUT BUFFER STATUS  BRANCH IF BUFFER EMPTY - CLOSE FILE IF EMPTY  SAVE (X) AS CURRENT INPUT POINTER - THIS WILL  ENABLE THE LIVE KEYBOARD (DIRECT) MODE. THE  LINE JUST ENTERED WILL BE INTERPRETED  GET NEXT CHARACTER FROM BASIC  NO LINE INPUT - GET ANOTHER LINE  BRANCH IF NUMBER C - THERE WAS A LINE NUMBER BEFORE  THE STATEMENT ENTERED, SO THIS STATEMENT

1781	AC96	ØD 61			TST	DEVNUM	* CHECK DEVICE NUMBER AND
1782		26 A			BNE	LAC46	* ISSUE DS ERROR IF DEVNUM ⇔ Ø
1783		BD B8			JSR	LB821	GO CRUNCH LINE
1784	AC9D	7E A	) CØ	*	JMP	LADCØ	GO EXECUTE THE STATEMENT (LIVE KEYBOARD)
1785 1786	ACAØ	A6 86	1	LACAØ	LDA	, χ+	GET A CHARACTER
1787		7E B9		Enons	JMP	LB9B1	SEND TO CONSOLE OUT
1788				* TAKE A L	INE FR	OM THE LINE INPUT BUFFER	
1789						INTO THE BASIC PROGRAM	
1790	ACA5	BD AF		LACA5	JSR	LAF67	CONVERT LINE NUMBER TO BINARY
1791 1792		9E 2E BF Ø2			LDX STX	BINVAL LINHDR	GET CONVERTED LINE NUMBER STORE IT IN LINE INPUT HEADER
1793		BD B8			JSR	LB821	GO CRUNCH THE LINE
1794	ACBØ	D7 Ø3	3		STB	TMPLOC	SAVE LINE LENGTH
1795	ACB2	8D 4E	)		BSR	LADØ1	FIND OUT WHERE TO INSERT LINE
1796	ACB4				BCS	LACC8	BRANCH IF LINE NUMBER DOES NOT ALREADY EXIST
1797 1798		DC 47			LDD	V47	GET ABSOLUTE ADDRESS OF LINE NUMBER
1799		D3 1E			SUBD ADDD	,X VARTAB	SUBTRACT ADDRESS OF NEXT LINE NUMBER * ADD TO CURRENT END OF PROGRAM - THIS WILL REMOVE
1800	ACBC	DD 1E			STD	VARTAB	* THE LENGTH OF THIS LINE NUMBER FROM THE PROGRAM
1801	ACBE	EE 84	1		LDU	, Х	POINT U TO ADDRESS OF NEXT LINE NUMBER
1802						E FROM BASIC PROGRAM	
1803	ACCØ	37 Ø2		LACCØ		A	GET A BYTE FROM WHAT S LEFT OF PROGRAM
1804 1805	ACC2 ACC4	A7 80			STA CMPX	,X+ VARTAB	MOVE IT DOWN COMPARE TO END OF BASIC PROGRAM
1806		26 F8			BNE	LACCØ	BRANCH IF NOT AT END
1807		B6 Ø2		LACC8		LINBUF	* CHECK TO SEE IF THERE IS A LINE IN
1808	ACCB	27 10			BEQ	LACE9	* THE BUFFER AND BRANCH IF NONE
1809	ACCD	DC 1E	3		LDD	VARTAB	= SAVE CURRENT END OF
1810	ACCF	DD 43			STD	V43	= PROGRAM IN V43
1811	ACD1	DB Ø3				TMPLOC	* ADD LENGTH OF CRUNCHED LINE,
1812 1813	ACD5	89 ØØ			ADCA STD	#Ø V41	* PROPOGATE CARRY AND SAVE NEW END * OF PROGRAM IN V41
1814	ACD7	BD AC			JSR	LAC1E	= MAKE SURE THERE S ENOUGH RAM FOR THIS
1815				*			= LINE & MAKE A HOLE IN BASIC FOR NEW LINE
1816	ACDA	CE Ø2	2 D8		LDU	#LINHDR-2	POINT U TO LINE TO BE INSERTED
1817	ACDD	37 Ø2		LACDD		A	GET A BYTE FROM NEW LINE
1818	ACDF	A7 80			STA	, X+	INSERT IT IN PROGRAM
1819 1820	ACE1 ACE3	9C 45			CMPX BNE	V45 LACDD	* COMPARE TO ADDRESS OF END OF INSERTED
1821		9E 41			LDX	V41	* LINE AND BRANCH IF NOT DONE = GET AND SAVE
1822		9F 1E			STX	VARTAB	= END OF PROGRAM
1823	ACE9	8D 36		LACE9	BSR	LAD21	RESET INPUT POINTER, CLEAR VARIABLES, INITIALIZE
1824	ACEB	8D Ø2	2		BSR	LACEF	ADJUST START OF NEXT LINE ADDRESSES
1825	ACED	20 80	)		BRA	LAC7C	REENTER BASIC S INPUT LOOP
1826	4055	05 1/				ART OF NEXT LINE ADDRESSES	
1827 1828		9E 19		LACEF LACF1	LDX LDD	TXTTAB ,X	POINT X TO START OF PROGRAM GET ADDRESS OF NEXT LINE
1829	ACF3	27 21		LACIT	BEQ	LAD16	RETURN IF END OF PROGRAM
1830	ACF5	33 Ø4			LEAU	4, X	POINT U TO START OF BASIC TEXT IN LINE
1831	ACF7	A6 C	ð	LACF7	LDA	,U+	* SKIP THROUGH THE LINE UNTIL A
1832	ACF9	26 F			BNE	LACF7	* ZERO (END OF LINE) IS FOUND
1833	ACFB	EF 84			STU	, X	SAVE THE NEW START OF NEXT LINE ADDRESS
1834 1835	ACFD ACFF	AE 84			LDX BRA	,X LACF1	POINT X TO START OF NEXT LINE KEEP GOING
1836	7.011		,	*	5101	2,101 2	NEEL GOING
1837				* FIND A L	INE NU	MBER IN THE BASIC PROGRAM	
1838					VITH CA	RRY SET IF NO MATCH FOUND	
1839	ADØ1			LADØ1	LDD	BINVAL	GET THE LINE NUMBER TO FIND
1840		9E 19		LADGE	LDX	TXTTAB	BEGINNING OF PROGRAM
1841 1842	ADØ5 ADØ7			LADØ5		, X LAD12	GET ADDRESS OF NEXT LINE NUMBER BRANCH IF END OF PROG
1843					CMPD		IS IT A MATCH?
1844						LAD14	CARRY SET IF LOWER; CARRY CLEAR IF MATCH
1845						, χ	X = ADDRESS OF NEXT LINE
1846						LADØ5	KEEP LOOPING FOR LINE NUMBER
1847				LAD12	ORCC		SET CARRY FLAG
1848 1849	AU14	9r 4/	'	LAD14	STX	V47	SAVE MATCH LINE NUMBER OR NUMBER OF LINE JUST AFTER WHERE IT SHOULD HAVE BEEN
1850	AD16	39		LAD16	RTS		1. 0.0000 0000
1851							
1852				* NEW			
1853				NEW	BNE	LAD14	BRANCH IF ARGUMENT GIVEN
1854						TXTTAB	GET START OF BASIC
1855 1856					CLR CLR		* PUT 2 ZERO BYTES THERE - ERASE * THE BASIC PROGRAM
1857							AND THE NEXT ADDRESS IS NOW THE END OF PROGRAM
1858				LAD21		TXTTAB	GET START OF BASIC
1859					JSR	LAEBB	PUT INPUT POINTER ONE BEFORE START OF BASIC
1860			_	* ERASE AL			
1861				LAD26			* RESET START OF STRING VARIABLES
1862							* TO TOP OF STRING SPACE
1863 1864						RESTOR VARTAB	RESET DATA POINTER TO START OF BASIC * GET START OF VARIABLES AND USE IT
1865							* TO RESET START OF ARRAYS
1866							RESET END OF ARRAYS
			L A9	LAD33			* RESET STRING STACK POINTER TO
1868						TEMPPT	* BOTTOM OF STRING STACK
1869	AD38	AE E4	1		LDX	,\$	GET RETURN ADDRESS OFF STACK

187Ø	AD3A	10 DE 21			LDS	FRETOP	RESTORE STACK POINTER
1871	AD3D	6F E2			CLR	,-S	PUT A ZERO BYTE ON STACK - TO CLEAR ANY RETURN OF
1872				*		,	FOR/NEXT DATA FROM THE STACK
1873	AD3F	ØF 2D			CLR	OLDPTR	RESET CONT ADDRESS SO YOU
1874	AD41	ØF 2E			CLR	OLDPTR+1	CAN T CONTINUE
1875 1876	AD43 AD45	ØF Ø8 6E 84			CLR JMP	ARYDIS	CLEAR THE ARRAY DISABLE FLAG RETURN TO CALLING ROUTINE - THIS IS NECESSARY
1877	AD45	UL 04		*	UNF	, Х	SINCE THE STACK WAS RESET
1878				*			ornor the other mile reserv
1879				* FOR			
1880				*			
1881						ND WILL STORE 18 BYTES ON	
1882 1883						LOOP WHICH IS BEING PROCE INED AS FOLLOWS: Ø- \$80 (F	
1884						ARIABLE DESCRIPTOR POINTER	
1885				,		RECTION: \$FF IF NEGATIVE;	
1886						E OF TO PARAMETER;	
1887				* 14,15=C	URRENT	LINE NUMBER; 16,17=RAM AD	DRESS OF THE END
1888						LINE CONTAINING THE FOR	
1889	AD47	86 80		FOR	LDA	#\$80	* SAVE THE DISABLE ARRAY FLAG IN VO8
189Ø 1891	AD49 AD4B	97 Ø8 BD AF 89			STA JSR	ARYDIS LET	* DO NOT ALLOW THE INDEX VARIABLE TO BE AN ARRAY SET INDEX VARIABLE TO INITIAL VALUE
1892	AD4B	BD AF 69			JSR	LABF9	SEARCH THE STACK FOR FOR/NEXT DATA
1893	AD51	32 62				2,5	PURGE RETURN ADDRESS OFF OF THE STACK
1894	AD53	26 Ø4			BNE	LAD59	BRANCH IF INDEX VARIABLE NOT ALREADY BEING USED
1895	AD55	9E ØF			LDX	TEMPTR	GET (ADDRESS + 18) OF MATCHED FOR/NEXT DATA
1896	AD57	32 85			LEAS		MOVE THE STACK POINTER TO THE BEGINNING OF THE
1897						NEXT DATA SO THE NEW DATA	
1898 1899						LD DATA. THIS WILL ALSO DE: ETURN AND FOR/NEXT DATA	
1900						THE STACK	BLEON
1901	AD59	C6 Ø9		LAD59	LDB	#\$Ø9	* CHECK FOR ROOM FOR 18 BYTES
1902	AD5B	BD AC 33			JSR	LAC33	* IN FREE RAM
1903	AD5E	BD AE E8			JSR	LAEE8	GET ADDR OF END OF SUBLINE IN X
1904	AD61	DC 68			LDD	CURLIN	GET CURRENT LINE NUMBER
1905	AD63	34 16			PSHS	X,B,A	SAVE LINE ADDR AND LINE NUMBER ON STACK
1906 1907	AD65 AD67	C6 A5 BD B2 6F			LDB JSR	#\$A5 LB26F	TOKEN FOR TO SYNTAX CHECK FOR TO
1907	AD67	BD B2 07			JSR	LB143	TM ERROR IF INDEX VARIABLE SET TO STRING
1909	AD6D	BD B1 41			JSR	LB141	EVALUATE EXPRESSION
1910				*			
1911	AD7Ø	D6 54			LDB	FPØSGN	GET FPAØ MANTISSA SIGN
1912	AD72	CA 7F			ORB	#\$7F	FORM A MASK TO SAVE DATA BITS OF HIGH ORDER MANTISSA
1913	AD74	D4 5Ø			ANDB	FPAØ	PUT THE MANTISSA SIGN IN BIT 7 OF HIGH ORDER MANTISSA
1914 1915	AD76 AD78	D7 50	7.5		STB LDY	FPAØ #LAD7E	SAVE THE PACKED HIGH ORDER MANTISSA
1916	AD7C	10 8E AD 7E B1 EA	7 F		JMP	#LAD7F LB1EA	LOAD FOLLOWING ADDRESS INTO Y AS A RETURN ADDRESS - PUSH FPAØ ONTO THE STACK
1917	AD7F	8E BA C5		LAD7F	LDX	#LBAC5	POINT X TO FLOATING POINT NUMBER 1.0 (DEFAULT STEP VALUE)
1918	AD82	BD BC 14			JSR	LBC14	MOVE (X) TO FPAØ
1919	AD85	9D A5			JSR	GETCCH	GET CURRENT INPUT CHARACTER
1920	AD87	81 A9			CMPA	#\$A9	STEP TOKEN
1921	AD89	26 Ø5			BNE	LAD90	BRANCH IF NO STEP VALUE
1922 1923	AD8B AD8D	9D 9F BD B1 41			JSR JSR	GETNCH LB141	GET A CHARACTER FROM BASIC EVALUATE NUMERIC EXPRESSION
1924	AD9Ø	BD BC 6D		LAD9Ø	JSR	LBC6D	CHECK STATUS OF FPAØ
1925	AD93	BD B1 E6			JSR	LB1E6	SAVE STATUS AND FPAØ ON THE STACK
1926	AD96	DC 3B			LDD	VARDES	* GET DESCRIPTOR POINTER FOR THE STEP
1927	AD98	34 Ø6			PSHS	В,А	* VARIABLE AND SAVE IT ON THE STACK
1928	AD9A	86 80			LDA	#\$80	= GET THE FOR FLAG AND
1929 1930	AD9C	34 Ø2			PSHS	A	= SAVE IT ON THE STACK
1930				* MAIN CO	MMAND	INTERPRETATION LOOP	
1932	AD9E	BD Ø1 9A		LAD9E	JSR .	RVEC2Ø	HOOK INTO RAM
1933		1C AF				#\$AF	ENABLE IRQ,FIRQ
1934		8D 46			BSR	LADEB	CHECK FOR KEYBOARD BREAK
1935		9E A6			LDX	CHARAD	GET BASIC S INPUT POINTER
1936		9F 2F			STX	TINPTR	SAVE IT
1937 1938		A6 8Ø 27 Ø7			LDA BEQ	,X+ LADB4	GET CURRENT INPUT CHAR & MOVE POINTER BRANCH IF END OF LINE
1939		81 3A			CMPA		CHECK FOR LINE SEPARATOR
1940		27 ØF			BEQ	LADCØ	BRANCH IF COLON
1941	ADB1	7E B2 77		LADB1	JMP	LB277	SYNTAX ERROR -IF NOT LINE SEPARATOR
1942		A6 81		LADB4	LDA	, χ++	GET MS BYTE OF ADDRESS OF NEXT BASIC LINE
1943	ADB6	97 ØØ			STA	ENDFLG	SAVE IN STOP/END FLAG - CAUSE A STOP IF
1944 1945				*			NEXT LINE ADDRESS IS < \$8000; CAUSE AN END IF ADDRESS > \$8000
1945	ADRR	27 5B			BEQ	LAE15	BRANCH TO STOP - END OF PROGRAM
1946		EC 8Ø			LDD	,X+	GET CURRENT LINE NUMBER
1948		DD 68			STD	CURLIN	SAVE IN CURLIN
1949	ADBE	9F A6			STX	CHARAD	SAVE ADDRESS OF FIRST BYTE OF LINE
1950		9D 9F		LADCØ	JSR	GETNCH	GET A CHARACTER FROM BASIC
1951		8D Ø2			BSR	LADC6	GO PROCESS COMMAND
1952 1953	ADC4 ADC6	2Ø D8 27 78		LADC6	BRA BEQ	LAD9E LAE4Ø	GO BACK TO MAIN LOOP RETURN IF END OF LINE
1953 1954	ADC8	27 78 4D		LAUCO	TSTA	LNEMU	CHECK FOR TOKEN - BIT 7 SET (NEGATIVE)
1955		10 2A 01	ВС		LBPL	LET	BRANCH IF NOT A TOKEN - GO DO A LET WHICH
1956				*	_		IS THE DEFAULT TOKEN FOR MICROSOFT BASIC
1957		81 A3			CMPA		SKIPF TOKEN - HIGHEST EXECUTABLE COMMAND IN BASIC
1958	ADCF	22 ØB			BHI	LADDC	BRANCH IF > A BASIC COMMAND

1959	ADD1	BE Ø1	23			LDX	COMVEC+3	GET ADDRESS OF BASIC S COMMAND TABLE
1960	ADD1	48	. 23		LADD4	ASLA	CONVECTS	X2 (2 BYTE/JUMP ADDRESS) & DISCARD BIT 7
1961	ADD5	1F 89	9		LADDY	TFR	A,B	SAVE COMMAND OFFSET IN ACCB
1962	ADD7	3A				ABX	,-	NON X POINTS TO COMMAND JUMP ADDR
1963	ADD8	9D 9F	=			JSR	GETNCH	GET AN INPUT CHAR
1964					*			
1965					* HERE IS		WE BRANCH TO DO A COMMANI	
1966		6E 94				JMP	[,X]	GO DO A COMMAND
1967	ADDC	81 B4			LADDC	CMPA		\$B4 IS HIGHEST BASIC TOKEN
1968 1969		23 D1 6E 9F		20		BLS JMP	LADB1 [COMVEC+13]	SYNTAX ERROR IF NON-EXECUTABLE TOKEN JUMP TO AN EX BAS COMMAND
1970	ADLE	0L 31	ют	20	*	UHI	[CONVECTIO]	CONT TO AN EX DAS COMMAND
1971					* RESTORE			
1972	ADE4	9E 19	9		RESTOR	LDX	TXTTAB	BEGINNING OF PROGRAM ADDRESS
1973	ADE6	3Ø 1F	=			LEAX	-1,X	MOVE TO ONE BYTE BEFORE PROGRAM
1974	ADE8	9F 33	3		LADE8	STX	DATPTR	SAVE NEW DATA POINTER
1975	ADEA	39				RTS		
1976					*			
1977	ADED	DD 41			* BREAK C		1.4101	CET A MENCEDOME ENTRY
1978 1979		BD A1			LADEB	JSR BEQ	LA1C1 LADFA	GET A KEYSTROKE ENTRY RETURN IF NO INPUT
1980		81 Ø3			LADFØ	CMPA		CONTROL C? (BREAK)
1981		27 15			27151 5	BEQ	STOP	YES
1982		81 13				CMPA		CONTROL S? (PAUSE)
1983	ADF6	27 Ø3	3			BEQ	LADFB	YES
1984	ADF8	97 87	7			STA	IKEYIM	SAVE KEYSTROKE IN INKEY IMAGE
1985	ADFA				LADFA	RTS		
1986		BD A1			LADFB		KEYIN	GET A KEY
1987		27 FE					LADER	BRANCH IF NO KEY DOWN
1988	ALUU	2Ø E	-		*	BRA	LADFØ	CONTINUE - DO A BREAK CHECK
1989 1990					* END			
1991	AFØ2	BD A4	1 26		END	JSR	LA426	CLOSE FILES
1992		9D AS				JSR	GETCCH	GET CURRENT INPUT CHAR
1993		20 02	2			BRA	LAEØB	
1994					*			
1995					* STOP			
1996		1A Ø1			STOP	ORCC		SET CARRY FLAG
1997		26 33			LAEØB	BNE	LAE4Ø	BRANCH IF ARGUMENT EXISTS
1998		9E A6				LDX	CHARAD	* SAVE CURRENT POSITION OF
1999		9F 2F				STX	TINPTR	* BASIC S INPUT POINTER
2000 2001		Ø6 ØØ			LAE11	ROR LEAS	ENDFLG	ROTATE CARRY INTO BIT 7 OF STOP/END FLAG PURGE RETURN ADDRESS OFF STACK
2001		9E 68			LAE15	LDX	CURLIN	GET CURRENT LINE NUMBER
2003		8C FF			LALIS		#\$FFFF	DIRECT MODE?
2004		27 Ø6					LAE22	YES
2005		9F 29					OLDTXT	SAVE CURRENT LINE NUMBER
2006	AE1E	9E 2F	-			LDX	TINPTR	* GET AND SAVE CURRENT POSITION
2007	AE2Ø	9F 2D	)			STX	OLDPTR	* OF BASIC S INPUT POINTER
2008		ØF 6F			LAE22	CLR	DEVNUM	SET DEVICE NUMBER TO SCREEN
2009		8E AE				LDX	#LABF2-1	POINT TO CR, BREAK MESSAGE
2010		ØD ØØ		4.0		TST	ENDFLG	CHECK STOP/END FLAG
2011 2012		10 2A				JMP	LAC73 LAC68	BRANCH TO MAIN LOOP OF BASIC IF END PRINT BREAK AT #### AND GO TO
2012	ALLU	/ L AC	, 00		*	UHI	LACOU	BASIC S MAIN LOOP IF STOP
2014								Short of intil 2001 II of or
2015					* CONT			
2016	AE3Ø	26 ØE	Ξ		CONT	BNE	LAE4Ø	RETURN IF ARGUMENT GIVEN
2017	AE32	C6 20	3			LDB	#2*16	CAN T CONTINUE ERROR
2018		9E 2E				LDX		GET CONTINUE ADDRESS (INPUT POINTER)
	AE36			ØC			LAC46	CN ERROR IF CONTINUE ADDRESS = Ø
2020		9F A6				STX	CHARAD	RESET BASIC S INPUT POINTER
2021		9E 29				LDX	OLDTXT	GET LINE NUMBER
2022 2023	AE4Ø	9F 68	5		LAE4Ø	STX RTS	CURLIN	RESET CURRENT LINE NUMBER
2023	A E 4 Ø	39			*	KIS		
2024					* CLEAR			
2026	AE41	27 20			CLEAR	BEQ	LAE6F	BRANCH IF NO ARGUMENT
2027		BD B3				JSR	LB3E6	EVALUATE ARGUMENT
2028	AE46	34 Ø6	5			PSHS	B,A	SAVE AMOUNT OF STRING SPACE ON STACK
2029		9E 27					MEMSIZ	GET CURRENT TOP OF CLEARED SPACE
2030		9D AS				JSR	GETCCH	GET CURRENT INPUT CHARACTER
2031		27 Ø0					LAE5A	BRANCH IF NO NEW TOP OF CLEARED SPACE
2032 2033		BD B2 BD B7				JSR JSR	LB26D LB73D	SYNTAX CHECK FOR COMMA EVALUATE EXPRESSION; RETURN VALUE IN X
2033		3Ø 1F					-1,X	X = TOP OF CLEARED SPACE
2035		9C 74					TOPRAM	COMPARE TO TOP OF RAM
2036		22 18				BHI	LAE72	OM ERROR IF > TOP OF RAM
2037		1F 10			LAE5A	TFR	X,D	ACCD = TOP OF CLEARED SPACE
2038		A3 E1				SUBD	, Š++	SUBTRACT OUT AMOUNT OF CLEARED SPACE
2039		25 12					LAE72	OM ERROR IF FREE MEM < Ø
2040		1F Ø3				TFR	D,U	U = BOTTOM OF CLEARED SPACE
2041		83 ØØ					#STKBUF	SUBTRACT OUT STACK BUFFER
2042		25 ØE				BCS	LAE72	OM ERROR IF FREE MEM < Ø
2043		93 1E					VARTAB	SUBTRACT OUT START OF VARIABLES
2044 2045		25 Ø7 DF 21				BCS STU	LAE72 FRETOP	OM ERROR IF FREE MEM < Ø SAVE NEW BOTTOM OF CLEARED SPACE
2045		9F 27				STX	MEMSIZ	SAVE NEW TOP OF CLEARED SPACE
2047		7E AD			LAE6F	JMP	LAD26	ERASE ALL VARIABLES, INITIALIZE POINTERS, ETC
								•

2048	AE72	7 F	Δ٢	44	LAE72	JMP	LAC44	OM ERROR
2049	AL/ L	, _	Α.		*	0111	ENOTT	OII ERROR
2050					* RUN			
2050	AE75	BD	a 1	9.1	RUN	JSR	RVEC18	HOOK INTO RAM
2052	AE78	BD			KUN	JSR	LA426	CLOSE ANY OPEN FILES
2053	AE7B	9 D				JSR	GETCCH	* GET CURRENT INPUT CHARACTER
2054	AE7D			FE AØ		LBEQ	LAD21	* IF NO LINE NUMBER
2055	AE81	BD				JSR	LAD26	ERASE ALL VARIABLES
2056	AE84	20				BRA	LAE9F	GOTO THE RUN ADDRESS
2057					*			****
2058					* G0			
2059	AE86	1F	89		GO	TFR	A,B	SAVE INPUT CHARACTER IN ACCB
2060	AE88	9 D			LAE88	JSR	GETNCH	GET A CHARACTER FROM BASIC
2061	AE8A	C1	A5			CMPB	#\$A5	TO TOKEN
2062	AE8C	27	16			BEQ	LAEA4	BRANCH IF GOTO
2063	AE8E	C1	Α6			CMPB	#\$A6	SUB TOKEN
2064	AE9Ø	26	45			BNE	LAED7	SYNTAX ERROR IF NEITHER
2065	AE92	C6				LDB	#3	=ROOM FOR 6
2066	AE94	BD	AC	33		JSR	LAC33	=BYTES ON STACK?
2067	AE97	DE	Α6			LDU	CHARAD	* SAVE CURRENT BASIC INPUT POINTER, LINE
2068	AE99	9 E	68			LDX	CURLIN	* NUMBER AND SUB TOKEN ON STACK
2069	AE9B	86	Α6			LDA	#\$A6	*
2070	AE9D	34	52			PSHS	U,X,A	*
2071	AE9F	8D	Ø3		LAE9F	BSR	LAEA4	GO DO A GOTO
2072	AEA1	7 E	ΑD	9E		JMP	LAD9E	JUMP BACK TO BASIC S MAIN LOOP
2073					* G0T0			
2074	AEA4	9 D	Α5		LAEA4	JSR	GETCCH	GET CURRENT INPUT CHAR
2075	AEA6	BD	ΑF	67		JSR	LAF67	GET LINE NUMBER TO BINARY IN BINVAL
2076	AEA9	8D	40			BSR	LAEEB	ADVANCE BASIC S POINTER TO END OF LINE
2077	AEAB	30	01			LEAX	\$01,X	POINT TO START OF NEXT LINE
2078	AEAD	DC	2B			LDD	BINVAL	GET THE LINE NUMBER TO RUN
2079	AEAF	10		68		CMPD	CURLIN	COMPARE TO CURRENT LINE NUMBER
2080	AEB2	22	Ø2			BHI	LAEB6	IF REO D LINE NUMBER IS > CURRENT LINE NUMBER,
2081					*			DON T START LOOKING FROM
2082					*			START OF PROGRAM
2083	AEB4	9 E				LDX	TXTTAB	BEGINNING OF PROGRAM
2084	AEB6	BD		05	LAEB6	JSR	LADØ5	GO FIND A LINE NUMBER
2085	AEB9	25				BCS	LAED2	UNDEFINED LINE NUMBER
2086	AEBB	30			LAEBB	LEAX	-1,X	MOVE BACK TO JUST BEFORE START OF LINE
2087	AEBD	9 F	A6			STX	CHARAD	RESET BASIC S INPUT POINTER
2088	AEBF	39			LAEBF	RTS		
2089					*			
2090					* RETURN			EVIT POURTE TE 120000EUT 010EU
2091	AECØ	26			RETURN	BNE	LAEBF	EXIT ROUTINE IF ARGUMENT GIVEN
2092	AEC2	86				LDA	#\$FF	* PUT AN ILLEGAL VARIABLE NAME IN FIRST BYTE OF
2093	AEC4	97	38			STA	VARDES	* VARDES WHICH WILL CAUSE FOR/NEXT DATA ON THE
2094	A F.C.C	D.D.	A D	F0	*	100	LAREO	STACK TO BE IGNORED
2095 2096	AEC6 AEC9	BD 1F		F 9		JSR TFR	LABF9	CHECK FOR RETURN DATA ON THE STACK RESET STACK POINTER - PURGE TWO RETURN ADDRESSES
2097	ALUS	ΙΓ	14			IFK	X,S	FROM THE STACK
2097	AECB	81	26			CMPA	#\$A6-\$8Ø	SUB TOKEN - \$80
2099	AECD	27				BEQ	LAEDA	BRANCH IF RETURN FROM SUBROUTINE
2100	AECF	C6				LDB	#2*2	ERROR #2 RETURN WITHOUT GOSUB
2101	AED1	80	D-1			FCB	SKP2	SKIP TWO BYTES
2102	AED2	C6	ØЕ		LAED2	LDB	#7*2	ERROR #7 UNDEFINED LINE NUMBER
2103	AED4	7 E		46	LALDE	JMP	LAC46	JUMP TO ERROR HANDLER
2104	AED7	7 E			LAED7	JMP	LB277	SYNTAX ERROR
2105	AEDA	35			LAEDA	PULS	A,X,U	* RESTORE VALUES OF CURRENT LINE NUMBER AND
2106	AEDC	9 F				STX	CURLIN	* BASIC S INPUT POINTER FOR THIS SUBROUTINE
2107	AEDE	DF	A6			STU	CHARAD	* AND LOAD ACCA WITH SUB TOKEN (\$A6)
2108					*			
2109					* DATA			
2110	AEEØ	8D	Ø6		DATA	BSR	LAEE8	MOVE INPUT POINTER TO END OF SUBLINE OR LINE
2111	AEE2	80				FCB	SKP2	OUTP O BUTTO
2112						1 00		SKIP 2 BYTES
						100	O	SKIP 2 BYTES
2113					* REM, EL		J	2KIN S BALES
2113					* REM, EL ELSE		5.H.Z	ZKIP Z BYIES
2114	AEE3	8D	Ø6			SE	LAEEB	MOVE INPUT POINTER TO END OF LINE
2114 2115 2116	AEE5	9 F			ELSE	SE		
2114 2115 2116 2117		9 F			ELSE REM LAEE7	BSR STX RTS	LAEEB CHARAD	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER
2114 2115 2116 2117 2118	AEE5 AEE7	9F 39	A6		ELSE REM LAEE7 * ADVANCE	SE BSR STX RTS INPUT	LAEEB CHARAD POINTER TO END OF SUBLIN	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER E OR LINE
2114 2115 2116 2117 2118 2119	AEE5 AEE7 AEE8	9F 39 C6	A6		ELSE REM  LAEE7 * ADVANCE LAEE8	BSR STX RTS INPUT LDB	LAEEB CHARAD POINTER TO END OF SUBLIN #':	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER
2114 2115 2116 2117 2118 2119 2120	AEE5 AEE7	9F 39 C6	A6		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA	BSR STX RTS INPUT LDB FCB	LAEEB CHARAD POINTER TO END OF SUBLIN #': SKP1LD	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F
2114 2115 2116 2117 2118 2119 2120 2121	AEE5 AEE7 AEE8	9F 39 C6	A6		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE	BSR STX RTS INPUT LDB FCB BASIC	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKPILD S INPUT POINTER TO END 0	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F
2114 2115 2116 2117 2118 2119 2120 2121 2122	AEE5 AEE7 AEE8 AEEA	9F 39 C6 86	A6		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE -	BSR STX RTS INPUT LDB FCB BASIC RETURN	LAEEB CHARAD POINTER TO END OF SUBLIN #': SKP1LD	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F IN X
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123	AEE5 AEE7 AEE8 AEEA	9F 39 C6 86	A6 3A		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB	LAEEB CHARAD  POINTER TO END OF SUBLING #': SKP1LD S INPUT POINTER TO END O ADDRESS OF END OF LINE+1	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124	AEE5 AEE7 AEE8 AEEA AEEA	9F 39 C6 86	A6 3A		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE -	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKPILD S INPUT POINTER TO END 0	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125	AEE5 AEE7 AEE8 AEEA AEEB AEEC AEEE	9F 39 C6 86 5F D7 5F	A6 3A Ø1		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE -	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB	LAEEB CHARAD  POINTER TO END OF SUBLIN #: SKP1LD S INPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR.
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126	AEE5 AEE7 AEE8 AEEA AEEB AEEC AEEE AEEF	9F 39 C6 86 5F D7 5F 9E	A6 3A Ø1 A6		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKP1LD S INPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC CHARAD	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127	AEE5 AEE7 AEE8 AEEA AEEB AEEC AEEE AEEF AEF1	9F 39 C6 86 5F D7 5F 9E 1F	A6 3A Ø1 A6 98		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE -	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX TFR	LAEEB CHARAD  POINTER TO END OF SUBLINE #': SKP1LD S INPUT POINTER TO END OF ADDRESS OF END OF LINE+1 CHARAC CHARAD B,A	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128	AEE5 AEE7 AEE8 AEEA AEEB AEEC AEEE AEEF AEF1	9F 39 C6 86 5F D7 5F 9E 1F	A6 3A Ø1 A6 98		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKP1LD S INPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC CHARAD	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCB TO CHARACTER  * FROM ACCB TO CHARACT - SAVE OLD TERMINATOR
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129	AEE5 AEE7 AEE8 AEEA AEEB AEEC AEEE AEEF AEF1 AEF3	9F 39 C6 86 5F D7 5F 9E 1F D6	A6 3A Ø1 A6 98 Ø1		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX TFR LDB	LAEEB CHARAD  POINTER TO END OF SUBLINE*: SKP1LD SINPUT POINTER TO END OF ADDRESS OF END OF LINE+1 CHARAC CHARAD B,A CHARAC	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCB TO CHARAC SAVE OLD TERMINATOR IN CHARAC
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130	AEE5 AEE7 AEE8 AEEA AEEB AEEC AEEC AEEF AEF1 AEF3 AEF5	9F 39 C6 86 5F D7 5F 9E 1F D6	A6 3A Ø1 A6 98 Ø1 Ø1		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB  LAEE1 *	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX TFR LDB	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKPILD S INPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC CHARAD B,A CHARAC CHARAC	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCE TO CHARAC - SAVE OLD TERMINATOR IN CHARAC SWAP PRIMARY AND SECONDARY TERMINATORS
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131	AEE5 AEE7 AEE8 AEEA AEEB AEEC AEEF AEF1 AEF3 AEF5 AEF7	9F 39 C6 86 5F D7 5F 9E 1F D6	A6 3A Ø1 A6 98 Ø1 Ø1 84		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB CLRB LDX TFR LDB STA LDA	LAEEB CHARAD  POINTER TO END OF SUBLIN! #': SKP11D S INPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC CHARAD B,A CHARAC CHARAC ,X	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCB TO CHARAC - SAVE OLD TERMINATOR IN CHARAC SWAP PRIMARY AND SECONDARY TERMINATORS GET NEXT INPUT CHARACTER
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2131 2131	AEES AEEA  AEEB AEEC AEEE AEEF AEF1 AEF3  AEF5 AEF7 AEF9	9F 39 C6 86 5F D7 5F 9E 1F D6	A6 3A Ø1 A6 98 Ø1 Ø1 84 EC		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB  LAEE1 *	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX TFR LDB STA LDA BEQ	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKP1LD SINPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC CHARAD B,A CHARAC CHARAC CHARAC ,X LAEE7	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCB TO CHARAC - SAVE OLD TERMINATOR IN CHARAC SWAP PRIMARY AND SECONDARY TERMINATORS GET NEXT INPUT CHARACTER RETURN IF Ø (END OF LINE)
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2132	AEES AEEA  AEEB AEEC AEEE AEEF AEF1 AEF3  AEF5 AEF7 AEF9 AEFB	9F 39 C6 86 5F D7 5F 9E 1F D6 97 A6 27 34	A6 3A Ø1 A6 98 Ø1 Ø1 84 EC Ø4		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB  LAEE1 *	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX TFR LDB STA LDA BEQ PSHS	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKPILD S INPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC CHARAD B,A CHARAC CHARAC ,X LAEE7 B	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCB TO CHARAC - SAVE OLD TERMINATOR IN CHARAC SWAP PRIMARY AND SECONDARY TERMINATORS GET NEXT INPUT CHARACTER RETURN IF Ø (END OF LINE) SAVE TERMINATOR ON STACK
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134	AEES AEEA  AEEB AEEC AEEE AEEF AEF1 AEF3  AEF5 AEF7 AEF9 AEFB AEFD	9F 39 C6 86 86 5F D7 5F 9E 1F D6 97 A6 27 34 A1	A6 3A Ø1 A6 98 Ø1 Ø1 84 EC Ø4 EØ		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB  LAEE1 *	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX TFR LDB STA LDA BEQ PSHS CMPA	LAEEB CHARAD  POINTER TO END OF SUBLIN #': SKPILD S INPUT POINTER TO END O ADDRESS OF END OF LINE+1 CHARAC CHARAD B,A CHARAC CHARAC ,X LAEE7 B	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  F IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCB TO CHARAC - SAVE OLD TERMINATOR IN CHARAC SWAP PRIMARY AND SECONDARY TERMINATORS GET NEXT INPUT CHARACTER RETURN IF Ø (END OF LINE) SAVE TERMINATOR ON STACK COMPARE TO INPUT CHARACTER
2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2132	AEES AEEA  AEEB AEEC AEEE AEEF AEF1 AEF3  AEF5 AEF7 AEF9 AEFB	9F 39 C6 86 86 5F D7 5F 9E 1F D6 27 34 A1 27	A6 3A Ø1 A6 98 Ø1 Ø1 84 EC Ø4 EØ E6		ELSE REM  LAEE7 * ADVANCE LAEE8 LAEEA * ADVANCE * LINE - LAEEB  LAEE1 *	BSR STX RTS INPUT LDB FCB BASIC RETURN CLRB STB CLRB LDX TFR LDB STA LDA BEQ PSHS	LAEEB CHARAD  POINTER TO END OF SUBLIN #: SKPILD SINPUT POINTER TO END O ADDRESS OF END OF LINE+1  CHARAC CHARAC B,A CHARAC CHARAC ,X LAEE7 B ,S+ LAEE7	MOVE INPUT POINTER TO END OF LINE RESET BASIC S INPUT POINTER  E OR LINE COLON = SUBLINE TERMINATOR CHARACTER SKPILD SKIP ONE BYTE; LDA #\$5F  IN X Ø = LINE TERMINATOR CHARACTER TEMP STORE PRIMARY TERMINATOR CHARACTER Ø (END OF LINE) = ALTERNATE TERM. CHAR. LOAD X W/BASIC S INPUT POINTER * CHANGE TERMINATOR CHARACTER * FROM ACCB TO CHARAC - SAVE OLD TERMINATOR IN CHARAC SWAP PRIMARY AND SECONDARY TERMINATORS GET NEXT INPUT CHARACTER RETURN IF Ø (END OF LINE) SAVE TERMINATOR ON STACK

2137	AFØ3	81 22			CMPA	#'"	CHECK FOR DOUBLE QUOTES
2138	AFØ5	27 EA			BEQ	LAEF1	BRANCH IF " - TOGGLE TERMINATOR CHARACTERS
2139	AFØ7	4 C			INCA		* CHECK FOR \$FF AND BRANCH IF
2140	AFØ8	26 Ø2			BNE	LAFØC	* NOT SECONDARY TOKEN
2141	AFØA	30 01			LEAX		MOVE INPUT POINTER 1 MORE IF SECONDARY
2142		81 86		LAFØC	CMPA	#\$85+1	TOKEN FOR IF?
2143 2144	AFIØ AFIØ	26 E7 ØC Ø4			BNE INC	LAEF7 IFCTR	NO - GET ANOTHER INPUT CHARACTER
2144	AFID	ØC Ø4		*	INC	IFCIR	INCREMENT IF COUNTER - KEEP TRACK OF HOW MANY IF STATEMENTS ARE NESTED IN ONE LINE
2146	AF12	2Ø E3			BRA	LAEF7	GET ANOTHER INPUT CHARACTER
2147							
2148				* IF			
2149	AF14	BD B1	41	IF	JSR	LB141	EVALUATE NUMERIC EXPRESSION
2150	AF17	9D A5			JSR	GETCCH	GET CURRENT INPUT CHARACTER
2151	AF19	81 81			CMPA	#\$81	TOKEN FOR GO
2152		27 Ø5			BEQ	LAF22	TREAT GO THE SAME AS THEN
2153 2154	AF1D AF1F	C6 A7 BD B2			LDB JSR	#\$A7 LB26F	TOKEN FOR THEN DO A SYNTAX CHECK ON ACCB
2154	AF22	96 4F		LAF22	LDA	FPØEXP	CHECK FOR TRUE/FALSE - FALSE IF FPAØ EXPONENT = ZERO
2156	AF24	26 13		2, 22	BNE	LAF39	BRANCH IF CONDITION TRUE
2157		ØF Ø4			CLR	IFCTR	CLEAR FLAG - KEEP TRACK OF WHICH NESTED ELSE STATEMENT
2158				*			TO SEARCH FOR IN NESTED IF LOOPS
2159	AF28	8D B6		LAF28	BSR	DATA	MOVE BASIC S POINTER TO END OF SUBLINE
2160	AF2A	4 D			TSTA		* CHECK TO SEE IF END OF LINE OR SUBLINE
2161	AF2B	27 BA			BEQ	LAEE7	* AND RETURN IF END OF LINE
2162 2163	AF2D	9D 9F			JSR	GETNCH #\$84	GET AN INPUT CHARACTER FROM BASIC
2164	AF2F AF31	81 84 26 F5			CMPA BNE	LAF28	TOKEN FOR ELSE IGNORE ALL DATA EXCEPT ELSE UNTIL
2165	AI JI	20 13		*	DITE	EN 20	END OF LINE (ZERO BYTE)
2166	AF33	ØA Ø4			DEC	IFCTR	CHECK TO SEE IF YOU MUST SEARCH ANOTHER SUBLINE
2167	AF35	2A F1			BPL	LAF28	BRANCH TO SEARCH ANOTHER SUBLINE FOR ELSE
2168	AF37	9D 9F			JSR	GETNCH	GET AN INPUT CHARACTER FROM BASIC
2169	AF39			LAF39	JSR	GETCCH	GET CURRENT INPUT CHARACTER
2170	AF3B		FF 65		LBCS	LAEA4	BRANCH TO GOTO IF NUMERIC CHARACTER
2171 2172	AF3F	7E AD	Cb		JMP	LADC6	RETURN TO MAIN INTERPRETATION LOOP
2172				* ON			
2174	AF42	BD B7	ØB	ON	JSR	LB7ØB	EVALUATE EXPRESSION
2175	AF45	C6 81			LDB	#\$81	TOKEN FOR GO
2176	AF47	BD B2	6 F		JSR	LB26F	SYNTAX CHECK FOR GO
2177	AF4A	34 Ø2			PSHS	A	SAVE NEW TOKEN (TO,SUB)
2178	AF4C	81 A6			CMPA	#\$A6	TOKEN FOR SUB?
2179		27 Ø4			BEQ	LAF54	YES
2180	AF5Ø	81 A5				#\$A5	TOKEN FOR TO?
2181 2182	AF52 AF54	26 83		LAF52	BNE DEC	LAED7 FPAØ+3	SYNTAX ERROR IF NOT SUB OR TO DECREMENT IS BYTE OF MANTISSA OF FPAØ – THIS
2183	AF34	ØA 53		LAF54 *	DEC	FFAU+3	IS THE ARGUMENT OF THE ON STATEMENT
2184	AF56	26 Ø5			BNE	LAF5D	BRANCH IF NOT AT THE PROPER GOTO OR GOSUB LINE NUMBER
2185	AF58	35 Ø4			PULS		GET BACK THE TOKEN FOLLOWING GO
2186	AF5A	7E AE			JMP	LAE88	GO DO A GOTO OR GOSUB
2187	AF5D	9D 9F		LAF5D	JSR	GETNCH	GET A CHARACTER FROM BASIC
2188	AF5F	8D Ø6			BSR	LAF67	CONVERT BASIC LINE NUMBER TO BINARY
2189		81 2C			CMPA		IS CHARACTER FOLLOWING LINE NUMBER A COMMA?
2190	AF63	27 EF			BEQ	LAF54	YES IF NOT, FALL THROUGH TO NEXT COMMAND
2191 2192	AF65 AF67	35 84			PULS	B,PC	
2193				I A E 6 7			
2194	AF69			LAF67	LDX	ZERO RINVAL	DEFAULT LINE NUMBER OF ZERO
	AF69	9F 2B		LAF67 *	STX	ZERO BINVAL	
2195	AF69			*	STX		DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL
		9F 2B		* * CONVERT	STX LINE	BINVAL NUMBER TO BINARY - RETURN	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL N VALUE IN BINVAL
2195 2196 2197	AF6B	9F 2B 24 61		* * CONVERT	STX LINE   BCC	BINVAL NUMBER TO BINARY - RETURN LAFCE	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL N VALUE IN BINVAL RETURN IF NOT NUMERIC CHARACTER
2195 2196 2197 2198	AF6B AF6D	9F 2B 24 61 8Ø 3Ø		* * CONVERT	STX LINE   BCC SUBA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  V VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII
2195 2196 2197 2198 2199	AF6B AF6D AF6F	9F 2B 24 61 8Ø 3Ø 97 Ø1		* * CONVERT	STX LINE   BCC SUBA STA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'Ø CHARAC	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1
2195 2196 2197 2198 2199 2200	AF6B AF6D AF6F AF71	9F 2B 24 61 8Ø 3Ø 97 Ø1 DC 2B		* * CONVERT	STX LINE BCC SUBA STA LDD	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1 GET ACCUMULATED LINE NUMBER VALUE
2195 2196 2197 2198 2199	AF6B AF6D AF6F AF71	9F 2B 24 61 8Ø 3Ø 97 Ø1		* * CONVERT	STX LINE   BCC SUBA STA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1
2195 2196 2197 2198 2199 2200 2201	AF6B AF6D AF6F AF71 AF73	9F 2B 24 61 8Ø 3Ø 97 Ø1 DC 2B		* * CONVERT	STX LINE BCC SUBA STA LDD CMPA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1 GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) -
2195 2196 2197 2198 2199 2200 2201 2202	AF6B AF6D AF6F AF71 AF73	9F 2B 24 61 8Ø 3Ø 97 Ø1 DC 2B 81 18		* * CONVERT	STX LINE   BCC SUBA STA LDD CMPA BHI	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  **VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER  MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE  LARGEST LINE NUMBER IS \$F9FF (63999) -  (24*256+255)*18+9
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205	AF6B AF6D AF6F AF71 AF73 AF75	9F 2B 24 61 80 30 97 01 DC 2B 81 18 22 DB		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206	AF6B AF6D AF6F AF71 AF73 AF75	9F 2B  24 61 80 30 97 01 DC 2B 81 18  22 DB  58 49		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  **VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1 GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256*255)*10*9 SYNTAX ERROR IF TOO BIG  * * TIMES 2
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79	9F 2B  24 61  80 30  97 01  DC 2B  81 18  22 DB  58  49  58		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256*255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 =
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A	9F 2B  24 61  8Ø 3Ø  97 Ø1  DC 2B  81 18  22 DB  58  49  58  49		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB ROLA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  * * TIMES 2 = TIMES 4
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B	9F 2B  24 61 80 30 97 01 DC 2B 81 18  22 DB  58 49 58 49 D3 2B		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB ROLA ADDD	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256*255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 =
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A	9F 2B  24 61 80 30 97 01 DC 2B 81 18  22 DB  58 49 D3 2B 58		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB ROLA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  **VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1 GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = TIMES 4 ADD 1 = TIMES 5
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2208 2209 2210	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7D AF7E	9F 2B  24 61 80 30 97 01 DC 2B 81 18  22 DB  58 49 D3 2B 58		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB ROLA ASLB ROLA ASLB ROLA ASLB ROLA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*18+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = = TIMES 4 ADD 1 = TIMES 5 *
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7B AF7E AF7F	9F 2B  24 61 8Ø 3Ø 97 Ø1 DC 2B 81 18  22 DB  58 49 D3 2B 58 49		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB ROLA ASLB ROLA ASLB ROLA ASLB ROLA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52 BINVAL CHARAC	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * * TIMES 10
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2210 2211 2212 2213 2214	AF6B AF6D AF6F AF71 AF73 AF75 AF78 AF79 AF7B AF7B AF7D AF7E AF7E AF81 AF83	24 61 80 30 97 01 DC 28 81 18 22 DB 58 49 58 49 D3 28 58 49 D8 01 89 00 DD 28		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB ROLA ADDD ASLB ROLA ADD ASLB ROLA ADDD ASLB ROL	BINVAL NUMBER TO BINARY - RETURN LAFCE #'Ø CHARAC BINVAL #24 LAF52 Ø BINVAL CHARAC ## BINVAL	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  * * TIMES 2 = = TIMES 4 ADD 1 = TIMES 5 * * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2211 2212 2211 2212 2213 2214	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7D AF7E AF7F AF81 AF83 AF85	24 61 80 30 97 01 DC 28 81 18 22 DB 58 49 58 49 58 49 D3 2B 58 49 D8 01 89 00 DD 2B 90 9F		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ROLA ASLB ROLA ASLB ROLA ADDD ASLB ROLA ADDB ADCA STD JSR	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52  BINVAL CHARAC #0 BINVAL BINVAL GETNCH	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2210 2211 2212 2213 2214 2215 2215 2216	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7D AF7E AF7F AF81 AF83 AF85	24 61 80 30 97 01 DC 28 81 18 22 DB 58 49 58 49 D3 28 58 49 D8 01 89 00 DD 28		* * CONVERT * LAF6B	BCC SUBA STA LDD CMPA BHI CD X 11 ROLA ASLB ROLA ASLB ROLA ADDD ASLB ROLA ADDB ADCA STD JSR	BINVAL NUMBER TO BINARY - RETURN LAFCE #'Ø CHARAC BINVAL #24 LAF52 Ø BINVAL CHARAC ## BINVAL	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  * * TIMES 2 = = TIMES 4 ADD 1 = TIMES 5 * * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2211 2212 2213 2214 2215 2216 2216 2217	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7D AF7E AF7F AF81 AF83 AF85	24 61 80 30 97 01 DC 28 81 18 22 DB 58 49 58 49 58 49 D3 2B 58 49 D8 01 89 00 DD 2B 90 9F		* * CONVERT * LAF6B  * * MULT AC	STX LINE   BCC SUBA STA LDD CMPA BHI CD X 11 ASLB ROLA ASLB ROLA ASLB ROLA ADDD ASLB ADDD ASLB BROLA BDD ADCA STD JSR BRA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52  BINVAL CHARAC #0 BINVAL BINVAL GETNCH	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2211 2212 2213 2214 2215 2216 2216	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7D AF7E AF7F AF81 AF83 AF85	24 61 80 30 97 01 DC 28 81 18 22 DB 58 49 58 49 58 49 D3 2B 58 49 D8 01 89 00 DD 2B 90 9F		* * CONVERT * LAF6B  * * MULT AC	STX  LINE   BCC SUBA LDD CMPA   BHI CD X 1  ASLB ROLA ASLB ROLA ADDD ROLA ASLB ROLA ADDD ASLB ROLA ADDD SERVICE CONTROL OF THE	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52 BINVAL CHARAC #8 BINVAL GETNCH LAF6B	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC
2195 2196 2197 2198 2199 2200 2201 2202 2203 2205 2206 2207 2208 2209 2211 2212 2213 2214 2215 2216 2217 2218 2216 2217 2218	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7D AF7E AF7F AF81 AF83 AF85	24 61 80 30 97 01 DC 28 81 18 22 DB 58 49 58 49 58 49 D3 2B 58 49 D8 01 89 00 DD 2B 90 9F		* CONVERT  * LAF6B  *  * MULT AC  *  * LET (EX  * EVALUAT	STX  LINE   BCC SUBA LDD CMPA  BHI CD X 11 ASLB ROLA ASLB ROLA ADDD ASLB ADCD JSR BRA  BRA BRA BRA BRA BRA BRA BRA BRA BRA BRA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52 BINVAL CHARAC #0 BINVAL CHARAC #0 BINVAL GETNCH LAF6B	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2210 2211 2212 2213 2214 2215 2216 2217 2216 2217 2218	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7B AF7B AF7B AF7B AF81 AF83 AF83 AF85 AF87	9F 2B  24 61 80 30 97 01 DC 2B 81 18  22 DB 58 49 03 2B 58 49 DD 2B 9D 9F 20 E2		* CONVERT  * LAF6B  * * MULT AC  * LET (EX * EVALUAT * TARGET	STX  LINE   BCC SUBA STA LDD CMPA BHI CD X II A SLB ROLA A ASLB ROLA ASLB ADDD ASLB ADDD ASLB BABA DE A STD JSR BABA DE A ROLA BABA DE A STD JSR BABA DE A ROLA BABA DE A R	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52  BINVAL CHARAC #0 BINVAL BINVAL GETNCH LAF6B	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1 GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  * * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC LOOP- PROCESS NEXT DIGIT
2195 2196 2197 2198 2199 2200 2201 2202 2203 2205 2206 2207 2208 2209 2211 2212 2213 2214 2215 2216 2217 2218 2216 2217 2218	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7B AF7B AF7B AF7B AF7B AF7B	24 61 80 30 97 01 DC 28 81 18 22 DB 58 49 58 49 58 49 D3 2B 58 49 D8 01 89 00 DD 2B 90 9F	57	* CONVERT  * LAF6B  *  * MULT AC  *  * LET (EX  * EVALUAT	STX  LINE   BCC SUBA STA LDD CMPA BHI CD X II A SLB ROLA A ASLB ROLA ASLB ADDD ASLB ADDD ASLB BABA DE A STD JSR BABA DE A ROLA BABA DE A STD JSR BABA DE A ROLA BABA DE A R	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52 BINVAL CHARAC #0 BINVAL CHARAC #0 BINVAL GETNCH LAF6B	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  *  * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF78 AF7B AF7B AF7E AF7E AF7E AF81 AF85 AF85 AF87	9F 2B  24 61 80 30 10 29 81 18  22 DB  58 49 D3 2B 58 49 D8 01 DB 01 DB 02 DB 02 DB 02 DB 02 DB 03 DB 03 DB 04 DB 05 DB	57	* CONVERT  * LAF6B  * * MULT AC  * LET (EX * EVALUAT * TARGET	STX  LINE   BCC SUBA STA LDD CMPA   BHI CD X 11 ASLB ROLA ASLB ROLA ASLB ROLA ADDD DSR BRA BADA BADA BADA BADA BADA BADA BADA	BINVAL NUMBER TO BINARY - RETURN LAFCE #'Ø CHARAC BINVAL #24 LAF52  BINVAL CHARAC #0 BINVAL GETNCH LAF6B	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  N VALUE IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9 SYNTAX ERROR IF TOO BIG  * * TIMES 2 = = TIMES 4 ADD 1 = TIMES 5 * * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC LOOP- PROCESS NEXT DIGIT
2195 2196 2197 2198 2199 22001 2202 2203 2204 2205 2206 2207 2208 2209 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7B AF7B AF7B AF83 AF83 AF85 AF87	9F 2B  24 61 80 30 97 01 DC 2B 81 18  22 DB 58 49 90 32 BB 83 90 9F 20 E2  BD 83 BD 83 BD 83 BD 82	57 6F	* CONVERT  * LAF6B  * * MULT AC  * LET (EX * EVALUAT * TARGET	STX  LINE   BCC SUBA STA LDD CMPA  BHI CD X II ASLB ROLA ASLB ROLA ASLB ROLA ASLB BROLA ADDB ADDB BRAD JSR  BBAS) BE A NOII = REPL JSR STX	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52 BINVAL CHARAC #0 BINVAL GETNCH LAF6B N-TOKEN EXPRESSION ACEMENT LB357 VARDES #\$B3 LB26F	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  **RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VOI  GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256+255)*10+9  SYNTAX ERROR IF TOO BIG  **  * TIMES 2 = = TIMES 4 ADD 1 = TIMES 5 *  * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC LOOP- PROCESS NEXT DIGIT  FIND TARGET VARIABLE DESCRIPTOR SAVE DESCRIPTOR ADDRESS OF 1ST EXPRESSION TOKEN FOR "=" DO A SYNTAX CHECK FOR =
2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2219 2210 2211 2212 2213	AF6B AF6D AF6F AF71 AF73 AF75 AF77 AF78 AF79 AF7A AF7B AF7B AF7B AF7B AF83 AF83 AF85 AF87	9F 28  24 61 80 30 97 01 DC 28 81 18  22 DB 58 49 90 32 85 89 90 9F 20 E2  BD 83 97 38 66 83	57 6F	* CONVERT  * LAF6B  * * MULT AC  * LET (EX * EVALUAT * TARGET	STX  LINE   BCC  BCC  SUBA  STA  LDD  CMPA  BHI  CD X II  ROLA  ASLB  ROLA  ASLB  ADDD  ASLB  ADDD  ADDA  ADDD  ADDA  BHA  DE  ADDB  ADCA  STD  JSR  BBA  BBAS)  E A NOI  E REPLI  JSR  STX  LDB	BINVAL NUMBER TO BINARY - RETURN LAFCE #'0 CHARAC BINVAL #24 LAF52 BINVAL CHARAC #0 BINVAL GETNCH LAF6B N-TOKEN EXPRESSION ACEMENT LB357 VARDES #\$B3	DEFAULT LINE NUMBER OF ZERO SAVE IT IN BINVAL  RETURN IF NOT NUMERIC CHARACTER MASK OFF ASCII SAVE DIGIT IN VO1 GET ACCUMULATED LINE NUMBER VALUE LARGEST LINE NUMBER IS \$F9FF (63999) - (24*256*255)*10*9 SYNTAX ERROR IF TOO BIG  * * TIMES 2 = TIMES 4 ADD 1 = TIMES 5 * * TIMES 10 ADD NEXT DIGIT PROPAGATE CARRY SAVE NEW ACCUMULATED LINE NUMBER GET NEXT CHARACTER FROM BASIC LOOP- PROCESS NEXT DIGIT  FIND TARGET VARIABLE DESCRIPTOR SAVE DESCRIPTOR ADDRESS OF 1ST EXPRESSION TOKEN FOR "="

2226	AF95	34	Ø2				PSHS	Α		* SAVE ON THE STACK
2227	AF97	BD	В1	56			JSR	LB156		EVALUATE EXPRESSION
2228	AF9A	35	Ø2				PULS	Α		* REGET VARIABLE TYPE OF 1ST EXPRESSION AND
2229	AF9C	46					RORA			* SET CARRY IF STRING
2230	AF9D	BD	В1	48			JSR	LB148		TYPE CHECK-TM ERROR IF VARIABLE TYPES ON
2231						*				BOTH SIDES OF EQUALS SIGN NOT THE SAME
2232	AFAØ	10	27	ØC	8F			LBC33		GO PUT FPAØ INTO VARIABLE DESCRIPTOR IF NUMERIC
2233									CRIPTOR IS LOCA	
2234									SPACE. TRANSFER	
2235									HE ADDRESS IN V	
2236									F IT IS ALREADY	
2237									SCRIPTOR FROM S	TRING
2238									ON THE STACK	
2239	AFA4	9E				LAFA4	LDX	FPAØ+2		POINT X TO DESCRIPTOR OF REPLACEMENT STRING
2240	AFA6	DC		~ ^			LDD	FRETOP		LOAD ACCD WITH START OF STRING SPACE
2241	AFA8	10		02			CMPD	2,X		IS THE STRING IN STRING SPACE?
2242	AFAB	24					BCC	LAFBE		BRANCH IF IT S NOT IN THE STRING SPACE
2243	AFAD	90					CMPX	VARTAB		COMPARE DESCRIPTOR ADDRESS TO START OF VARIABLES
2244	AFAF	25				LAED1	BCS	LAFBE		BRANCH IF DESCRIPTOR ADDRESS NOT IN VARIABLES
2245	AFB1	E6		αn		LAFB1	LDB	, X		GET LENGTH OF REPLACEMENT STRING RESERVE ACCB BYTES OF STRING SPACE
2246	AFB3 AFB6	9 E	B5	טש			JSR LDX	LB5ØD V4D		GET DESCRIPTOR ADDRESS BACK
2247 2248	AFB8	BD		12			JSR	LB643		MOVE STRING INTO STRING SPACE
2249	AFBB		ØØ				LDX	#STRDES		POINT X TO TEMP STRING DESCRIPTOR ADDRESS
2250	AFBE	9 F		50		LAFBE	STX	V4D		SAVE STRING DESCRIPTOR ADDRESS IN V4D
2251	AFCØ		B6	75		LAIDE	JSR	LB675		REMOVE STRING DESCRIPTOR IF LAST ONE
2252	AT CD	00	50	, ,		*	0310	LD073		ON STRING STACK
2253	AFC3	DE	4 N				LDU	V4D		POINT U TO REPLACEMENT DESCRIPTOR ADDRESS
2254	AFC5	9 E					LDX	VARDES		GET TARGET DESCRIPTOR ADDRESS
2255	AFC7	37					PULU	A , B , Y		GET LENGTH AND START OF REPLACEMENT STRING
2256	AFC9	A7					STA	, Х		* SAVE STRING LENGTH AND START IN
2257	AFCB		AF	Ø2			STY	2,X		* TARGET DESCRIPTOR LOCATION
2258	AFCE	39		-		LAFCE	RTS	-, ~		THIRD DECORPTION CONTINUE
2259										
2260	AFCF	3F	52	45	44 4F	LAFCF	FCC	'?REDO'		?REDO MESSAGE
2261	AFD4	ØD					FCB	CR,\$ØØ		
2262								ŕ		
2263	AFD6	C6	22			LAFD6	LDB	#2*17		BAD FILE DATA ERROR
2264	AFD8	ØD	6F				TST	DEVNUM		CHECK DEVICE NUMBER AND BRANCH
2265	AFDA	27	Ø3				BEQ	LAFDF		IF SET TO SCREEN
2266	AFDC	7 E	AC	46		LAFDC	JMP	LAC46		JMP TO ERROR HANDLER
2267	AFDF	96	Ø9			LAFDF	LDA	INPFLG		= GET THE INPUT FLAG AND BRANCH
2268	AFE1	27	Ø7				BEQ	LAFEA		= IF INPUT
2269	AFE3	9 E	31				LDX	DATTXT		* GET LINE NUMBER WHERE THE ERROR OCCURRED
2270	AFE5	9 F	68				STX	CURLIN		* AND USE IT AS THE CURRENT LINE NUMBER
2271	AFE7	7 E	В2	77			JMP	LB277		SYNTAX ERROR
2272	AFEA	8E	ΑF	CE		LAFEA	LDX	#LAFCF-1		* POINT X TO ?REDO AND PRINT
2273	AFED	BD	В9	9C			JSR	LB99C		* IT ON THE SCREEN
2274	AFFØ	9 E	2F				LDX	TINPTR		= GET THE SAVED ABSOLUTE ADDRESS OF
2275	AFF2	9 F	Α6				STX	CHARAD		= INPUT POINTER AND RESTORE IT
2276	AFF4	39					RTS			
2277						*				
2278						* INPUT				
2279	AFF5	C6				INPUT	LDB	#11*2		ID ERROR
2280	AFF7	9 E					LDX	CURLIN		GET CURRENT LINE NUMBER
2281	AFF9	30					LEAX	1,X		ADD ONE
2282	AFFB	27					BEQ	LAFDC		ID ERROR BRANCH IF DIRECT MODE
2283	AFFD		Ø3				BSR	LBØØ2		GET SOME INPUT DATA
2284 2285	AFFF BØØ1	ØF 39	6F				CLR RTS	DEVNUM		SET DEVICE NUMBER TO SCREEN
2286	BØØ2	81	23			LBØØ2	CMPA	#'#		CHECK FOR DEVICE NUMBER
2286	BØØ4	26				LDUNG	BNE	# # LBØØF		NO DEVICE NUMBER GIVEN
2288	BØØ6			۸۶			JSR	LA5A5		CHECK SYNTAX AND GET DEVICE NUMBER
2289	BØØ9						JSR	LA3A5 LA3ED		CHECK FOR VALID INPUT FILE
2290	BØØC						JSR	LB26D		SYNTAX CHECK FOR COMMA
2291	BØØF			- 5		LBØØF	CMPA	#'"		CHECK FOR PROMPT STRING DELIMITER
2292	BØ11						BNE	LBØ1E		BRANCH IF NO PROMPT STRING
2293	BØ13			44			JSR	LB244		PUT PROMPT STRING ON STRING STACK
2294	BØ16						LDB	#';		*
2295	BØ18			6F			JSR	LB26F		* DO A SYNTAX CHECK FOR SEMICOLON
2296	BØ1B						JSR	LB99F		PRINT MESSAGE TO CONSOLE OUT
2297	BØ1E	8E	Ø2	DC		LBØ1E	LDX	#LINBUF		POINT TO BASIC S LINE BUFFER
2298	BØ21	6F	84				CLR	, X		CLEAR 1ST BYTE - FLAG TO INDICATE NO DATA
2299						*				IN LINE BUFFER
2300	BØ23	ØD	6F				TST	DEVNUM		CHECK DEVICE NUMBER
2301	BØ25						BNE	LBØ49		BRANCH IF NOT SET TO SCREEN
2302	BØ27	8D					BSR	LBØ2F		INPUT A STRING TO LINE BUFFER
23Ø3	BØ29						LDB	#',		* INSERT A COMMA AT THE END
2304	BØ2B						STB	, Х		* OF THE LINE INPUT BUFFER
23Ø5	BØ2D	20	1A				BRA	LBØ49		
23Ø6									BUFFER CONSOLE	
23Ø7	BØ2F					LBØ2F	JSR	LB9AF		SEND A "?" TO CONSOLE OUT
23Ø8	BØ32						JSR	LB9AC		SEND A SPACE TO CONSOLE OUT
2309	BØ35			90		LBØ35	JSR	LA39Ø		GO READ IN A BASIC LINE
2310	BØ38						BCC	LBØ3F		BRANCH IF ENTER KEY ENDED ENTRY
2311	ВØЗА						LEAS			PURGE TWO RETURN ADDRESSES OFF THE STACK
2312	BØ3C	7 E		11			JMP	LAE11		GO DO A STOP IF BREAK KEY ENDED LINE ENTRY
2313		C6				LBØ3F	LDB	#2*23		INPUT PAST END OF FILE ERROR
2314	BØ41	ЙD	10				TST	CINBFL		CHECK FOR MORE CHARACTERS IN CONSOLE IN BUFFER

2315									
	BØ43	26	97				BNE	LAFDC	IE ERROR IF EMPTY
2316	BØ45	39					RTS		
2317	5543	33				*	1(13		
2318						* READ			
2319	BØ46	9 E	33			READ	LDX	DATPTR	GET READ START ADDRESS
2320	BØ48	86					FCB	SKP1LD	SKIP ONE BYTE - LDA #*\$4F
2321	BØØ49					LBØ49	CLRA		INPUT ENTRY POINT: INPUT FLAG = Ø
2322	BØ4A	97	Ø9				STA	INPFLG	SET INPUT FLAG; Ø = INPUT: <> Ø = READ
2323	BØ4C	9 F					STX	DATTMP	SAVE READ START ADDRESS/ INPUT BUFFER START
2324	BØ4E	BD		57		LBØ4E	JSR	LB357	EVALUATE A VARIABLE
2325	BØ51	9 F					STX	VARDES	SAVE DESCRIPTOR ADDRESS
2326	BØ53	9 E					LDX	CHARAD	* GET BASIC S INPUT POINTER
2327	BØ55	9 F					STX	BINVAL	* AND SAVE IT
2328	BØ57	9 E					LDX	DATTMP	GET READ ADDRESS START/ INPUT BUFFER POINTER
2329	BØ59	Α6	84				LDA	, χ	GET A CHARACTER FROM THE BASIC PROGRAM
2330	BØ5B	26	ØC				BNE	LBØ69	BRANCH IF NOT END OF LINE
2331	BØ5D	96					LDA	INPFLG	* CHECK INPUT FLAG AND BRANCH
2332	BØ5F	26	58				BNE	LBØB9	* IF LOOKING FOR DATA (READ)
2333						* NO DATA	IN IN	PUT LINE BUFFER AND/OR II	NPUT
2334						* NOT COM	ING FRO	DM SCREEN	
2335	BØ61	BD	Ø1	7 C			JSR	RVEC1Ø	HOOK INTO RAM IF INPUT
2336	BØ64	BD	В9	ΑF			JSR	LB9AF	SEND A '?' TO CONSOLE OUT
2337	BØ67	8D	С6				BSR	LBØ2F	FILL INPUT BUFFER FROM CONSOLE IN
2338	BØ69	9 F	Α6			LBØ69	STX	CHARAD	RESET BASIC S INPUT POINTER
2339	BØ6B	9 D	9 F				JSR	GETNCH	GET A CHARACTER FROM BASIC
2340	BØ6D	D6	Ø6				LDB	VALTYP	* CHECK VARIABLE TYPE AND
2341	BØ6F	27	27				BEQ	LBØ98	* BRANCH IF NUMERIC
2342						* READ/IN	PUT A S	STRING VARIABLE	
2343	BØ71	9 E					LDX	CHARAD	LOAD X WITH CURRENT BASIC INPUT POINTER
2344	BØ73	97					STA	CHARAC	SAVE CURRENT INPUT CHARACTER
2345	BØ75	81	22				CMPA	#'"	CHECK FOR STRING DELIMITER
2346	BØ77	27	12				BEQ	LBØ8B	BRANCH IF STRING DELIMITER
2347	BØ79	30	1 F				LEAX	-1,X	BACK UP POINTER
2348	BØ7B	4 F					CLRA		* ZERO = END OF LINE CHARACTER
2349	BØ7C	97	Ø1				STA	CHARAC	* SAVE AS TERMINATOR
2350	BØ7E	BD		5F			JSR	LA35F	SET UP PRINT PARAMETERS
2351	BØ81	ØD	6 E				TST	PRTDEV	CHECK PRINT DEVICE NUMBER
2352	BØ83	26	Ø6				BNE	LBØ8B	BRANCH IF CASSETTE - USE TWO ZEROS AS TERMINATOR
2353						*			CHARACTERS FOR CASSETTE
2354	BØ85	86	3 A				LDA	#':	END OF SUBLINE CHARACTER
2355	BØ87	97					STA	CHARAC	SAVE AS TERMINATOR I
2356	BØ89	86					LDA	#',	COMMA
2357	BØ8B	97				LBØ8B	STA	ENDCHR	SAVE AS TERMINATOR 2
2358	BØ8D	BD					JSR	LB51E	STRIP A STRING FROM THE INPUT BUFFER
2359	BØ9Ø	BD					JSR	LB249	MOVE INPUT POINTER TO END OF STRING
2360	BØ93	BD		Α4			JSR	LAFA4	PUT A STRING INTO THE STRING SPACE IF NECESSARY
2361	BØ96	20	Ø6				BRA	LBØ9E	CHECK FOR ANOTHER DATA ITEM
2362								C VALUE IN A READ OR INPUT	
2363	BØ98	BD				LBØ98	JSR	LBD12	CONVERT AN ASCII STRING TO FP NUMBER
2364	BØ9B	BD	ВС	33			JSR	LBC33	PACK FPAØ AND STORE IT IN ADDRESS IN VARDES -
						*			INPUT OR READ DATA ITEM
2365		9 D				LBØ9E	JSR	GETCCH	GET CURRENT INPUT CHARACTER
2366	BØ9E						BEQ	LBØA8	BRANCH IF END OF LINE
2366 2367	BØAØ	27					CMPA	#',	CHECK FOR A COMMA
2366 2367 2368	BØAØ BØA2	27 81	2C					LAFD6	
2366 2367 2368 2369	BØAØ BØA2 BØA4	27 81 10	2C 26	FF	2E	1.0040	LBNE	CHARAR	'BAD FILE DATA' ERROR OR RETRY
2366 2367 2368 2369 2370	BØAØ BØA2 BØA4 BØA8	27 81 10 9E	2C 26 A6	FF	2E	LBØA8	LDX	CHARAD	* GET CURRENT INPUT
2366 2367 2368 2369 2370 2371	BØAØ BØA2 BØA4 BØA8 BØAA	27 81 10 9E 9F	2C 26 A6 35	FF	2E	LBØA8	LDX STX	DATTMP	* GET CURRENT INPUT * POINTER (USED AS A DATA POINTER) AND SAVE IT
2366 2367 2368 2369 2370 2371 2372	BØAØ BØA2 BØA4 BØA8 BØAA	27 81 10 9E 9F 9E	2C 26 A6 35 2B	FF	2E	LBØA8	LDX STX LDX	DATTMP BINVAL	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR
2366 2367 2368 2369 2370 2371 2372 2373	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE	27 81 10 9E 9F 9E 9F	2C 26 A6 35 2B A6	FF	2E	LBØA8	LDX STX LDX STX	DATTMP BINVAL CHARAD	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT
2366 2367 2368 2369 2370 2371 2372 2373 2374	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ	27 81 10 9E 9F 9E 9F 9D	2C 26 A6 35 2B A6 A5	FF	2E	LBØA8	LDX STX LDX STX JSR	DATTMP BINVAL CHARAD GETCCH	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ BØBØ	27 81 10 9E 9F 9E 9F 9D 27	2C 26 A6 35 2B A6 A5		2E	LBØA8	LDX STX LDX STX JSR BEQ	DATTMP BINVAL CHARAD GETCCH LBØD5	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ BØB2 BØB4	27 81 10 9E 9F 9E 9F 9D 27 BD	2C 26 A6 35 2B A6 A5 21 B2		2E	LBØA8	LDX STX LDX STX JSR BEQ JSR	DATTMP BINVAL CHARAD GETCCH LB0D5 LB26D	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ BØBØ	27 81 10 9E 9F 9E 9F 9D 27 BD	2C 26 A6 35 2B A6 A5 21 B2		2E		LDX STX LDX STX JSR BEQ JSR BRA	DATTMP BINVAL CHARAD GETCCH LB0D5 LB26D LB24E	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ BØB2 BØB4	27 81 10 9E 9F 9E 9F 9D 27 BD	2C 26 A6 35 2B A6 A5 21 B2		2E	* SEARCH	LDX STX LDX STX JSR BEQ JSR BRA FROM AE	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB04E DDRESS IN X FOR	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ BØB2 BØB4 BØB7	27 81 10 9E 9F 9E 9F 9D 27 BD 20	2C 26 A6 35 2B A6 A5 21 B2 95		2E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AC	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LBØ4E ODRESS IN X FOR OF THE TOKEN FOR DATA	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ BØB2 BØB4 BØB7	27 81 10 9E 9F 9E 9F 9D 27 BD 20	2C 26 A6 35 2B A6 A5 21 B2 95	6D	2E	* SEARCH	LDX STX LDX STX JSR BEQ JSR BRA FROM AE	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LBØ4E ODRESS IN X FOR OF THE TOKEN FOR DATA CHARAD	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381	BØAØ BØA2 BØA4 BØA8 BØAA BØAC BØAE BØBØ BØB2 BØB4 BØB7	27 81 10 9E 9F 9D 27 BD 20	2C 26 A6 35 2B A6 A5 21 B2 95	6D	2E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AC URENCE STX JSR	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2380 2381 2382	BØAØ BØA2 BØA4 BØAA BØAA BØAE BØBØ BØB2 BØB4 BØB7	27 81 10 9E 9F 9D 27 BD 20 9F BD 30	2C 26 A6 35 2B A6 A5 21 B2 95	6D	2E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AE URENCE STX JSR LEAX	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER SEARCH FOR END OF CURRENT LINE OR SUBLINE MOVE X ONE PAST END OF LINE
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383	BØAØ BØA2 BØA4 BØAA BØAA BØAE BØBØ BØB2 BØB4 BØB7 BØB9 BØBB BØBB	27 81 10 9E 9F 9B 27 BD 20 9F 30 4D	2C 26 A6 35 2B A6 A5 21 B2 95 A6 AE Ø1	6D	2E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AD URENCE STX JSR LEAX TSTA	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LBØ4E DORESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEEB 1,X	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384	BØAØ BØA2 BØAA BØAA BØAC BØAE BØBØ BØB2 BØB4 BØB7	27 81 10 9E 9F 9F 9D 27 8D 20 9F 8D 30 4D 26	2C 26 A6 35 2B A6 A5 21 B2 95 A6 AE Ø1	6D	2E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AE URENCE STX JSR LEAX TSTA BNE	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E JDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEEB 1,X LBØCD	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385	BØAØ BØA2 BØAA BØAA BØAC BØAE BØBØ BØB2 BØB4 BØB7 BØB9 BØBB BØBB BØBB BØBCØ BØCØ BØCCI BØCC3	27 81 10 9E 9F 9F 9D 27 BD 20 9F 8D 30 4D 26 C6	2C 26 A6 35 2B A6 A5 21 B2 95 A6 AE Ø1	6D	2E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AE URENCE STX JSR LEAX TSTA BNE LDB	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1,X  LBØCD #2*3	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384	BØAØ BØA2 BØAA BØAA BØAC BØAE BØBØ BØB2 BØB4 BØB7	27 81 10 9E 9F 9D 27 BD 20 9F 8D 30 4D 26 C6 EE	2C 26 A6 35 2B A6 A5 21 B2 95 A6 AE Ø1 ØA Ø6 81	6D	22E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AE URENCE STX JSR LEAX TSTA BNE	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E JDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEEB 1,X LBØCD	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2389 2381 2382 2383 2384 2384 2386	BØAØ BØA2 BØAA BØAA BØAC BØAE BØBB BØB2 BØB4 BØB7 BØB9 BØBB BØBB BØBB BØBB BØCØ BØCØ BØCØ	27 81 10 9E 9F 9D 27 BD 20 9F BD 30 4D 26 C6 EE 27	2C 26 35 2B A6 A5 21 B2 95 A6 AE 01 0A 06 81 41	6D	22E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AC URENCE STX JSR LEAX TSTA BNE LDB LDU	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E JDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1,X LBØCD #2*3 ,X++ LB10A	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER SEARCH FOR END OF CURRENT LINE OR SUBLINE MOVE X ONE PAST END OF LINE CHECK FOR END OF LINE BRANCH IF END OF SUBLINE OUT OF DATA ERROR GET NEXT 2 CHARACTERS
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387	BØAØ BØA2 BØAA BØAA BØAC BØAE BØBØ BØB2 BØB4 BØB7 BØB9 BØBB BØBE BØCØ BØC1 BØC3 BØC5 BØC5	27 81 10 9E 9F 9D 27 BD 20 9F 8D 20 9F 20 6 6 6 6 6 6 6 7 7 8 7 8 7 8 8 8 8 8 8 8	2C 26 35 2B A6 21 B2 B2 B2 B2 B2 B4 B4 B4 B4 B4 B4 B4 B4 B4 B4 B4 B4 B4	6D	22E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AD URENCE STX JSR LEAX TSTA BNE LDB LDU BEQ	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1,X LBØCD #2*3 ,X++	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2381 2382 2383 2381 2382 2383 2384 2385 2386 2387 2388	BØAØ BØA2 BØAA BØAA BØAC BØAE BØB2 BØB4 BØB7 BØB9 BØBB BØB6 BØCØ BØC1 BØC3 BØC7 BØC7 BØC9	27 81 10 9E 9F 9B 27 BD 20 9F 8D 20 9F 8D 20 6 6 6 6 6 6 6 7 6 7 8 7 8 7 8 8 7 8 8 7 8 8 7 8 8 8 8	2C 26 A6 35 2B A6 A5 21 B2 95 A6 A8 A8 A8 A8 A8 A8 A8 A8 A8 A8 A8 A8 A8	6D	22E	* SEARCH * 1ST OCC	LDX STX LDX STX JSR BEQ JSR BRA FROM AD URENCE STX JSR LEAX TSTA BNE LDB LDU BEQ LDD	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEEB 1,X LBØCD #2*3 ,X++ LB10A ,X++	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2380 2381 2382 2383 2384 2385 2386 2387 2388	BØAØ BØA2 BØAA BØAA BØAC BØAB BØBØ BØBB BØBB BØBB	27 81 10 9E 9F 9D 27 8D 20 9F 8D 30 4D 26 6 EE 27 EC DD A6	2C 26 35 2B 2A 6 21 B2 95 A6 AE 07 41 81 81 84	6D	22E	* SEARCH * 1ST OCC LBØB9	LDX STX LDX STX JSR BEQ JSR BRA BRA URENCE STX JSR LEAX LEAX BNE LDB LDU BEQ LDD STD	DATTMP BINNAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E OF THE TOKEN FOR DATA CHARAD LAEE8 1, X LBØCD #2*3 ,X++ LB1ØA ,X++ DATTXT ,X	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER SEARCH FOR END OF CURRENT LINE OR SUBLINE MOVE X ONE PAST END OF LINE CHECK FOR END OF LINE BRANCH IF END OF SUBLINE OUT OF DATA ERROR GET NEXT 2 CHARACTERS OD ERROR IF END OF PROGRAM GET BASIC LINE NUMBER AND SAVE IT IN DATIXT
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2389	BØAØ BØA2 BØAA BØAA BØAA BØAE BØBØ BØB2 BØB4 BØB7 BØB9 BØBB BØBB BØCØ BØCC1 BØCC5 BØCC7 BØCC9 BØCCD	27 81 10 9E 9F 9E 9F 9D 27 8D 20 9F 8D 30 4D 26 6 EE 27 EC DD A6 81	2C 26 A6 35 2B A6 A21 B25 A6 AE1 ØA 681 481 886	6D	22E	* SEARCH * 1ST OCC LBØB9	LDX STX LDX STX JSR BEQ JSR BRA BRA URENCE STX JSR LEAX TSTA BNE LDB LDU BEQ LDD STD LDA	DATTMP BINNAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E OF THE TOKEN FOR DATA CHARAD LAEE8 1, X LBØCD #2*3 ,X++ LB1ØA ,X++ DATTXT ,X	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATTAT  GET AN INPUT CHARACTER
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2389 2389 2399 2390 2391	BØAØ BØAA2 BØAA8 BØAA BØAAC BØABE BØBØ BØBB2 BØBB4 BØBB7 BØBB BØBB BØBB BØC1 BØC3 BØC7 BØC9 BØC9 BØCB BØCB BØCB	27 81 10 9E 9F 9B 27 BD 20 9F 8D 26 C6 EE 27 EC DD A6 81 26	2C 26 A6 35 2B A6 A5 2B A6	6D	22E	* SEARCH * 1ST OCC LBØB9	LDX STX LDX STX JSR BEQ JSR BRA FROM AL URENCE STX JSR LEAX TSTA BNE LDB LDU BEQ LDD STD LDA CMPA	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEEB 1,X LBØCD #2*3 ,X++ LB10A ,X++ DATTXT ,X #\$86	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATIXT  GET AN INPUT CHARACTER  DATA TOKEN?
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2381 2382 2383 2384 2385 2385 2386 2387 2388 2389 2399 23991 2392	BØAØ BØA2 BØAA BØAA BØAC BØBØ BØB2 BØB4 BØB7 BØB5 BØB5 BØC1 BØC3 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5	27 81 10 9E 9F 9B 27 BD 20 9F 8D 26 C6 EE 27 EC DD A6 81 26	2C 26 A6 35 2B A6 A5 2B A6	6D	22E	* SEARCH * 1ST OCC LBØB9	LDX STX LDX STX JSR BEQ JSR BRA FROM AE URENCE STX JSR LEAX TSTA BNE LDB LDU BEQ LDD STD LDD CMPA BRA BRA	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1,X  LBØCD #2*3 ,X++ LB1ØA ,X++ DATTXT ,X #\$86 LBØB9	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER SEARCH FOR END OF CURRENT LINE OR SUBLINE MOVE X ONE PAST END OF LINE GHECK FOR END OF LINE BRANCH IF END OF SUBLINE OUT OF DATA ERROR GET NEXT 2 CHARACTERS OD ERROR IF END OF PROGRAM GET BASIC LINE NUMBER AND SAVE IT IN DATIXT GET AN INPUT CHARACTER DATA TOKEN? NO KEEP LOOKING
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2390 2391	BØAØ BØA2 BØAA BØAA BØAC BØBØ BØB2 BØB4 BØB7 BØB5 BØB5 BØC1 BØC3 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5	27 81 10 9E 9F 9F 9D 27 8D 20 9F 8D 20 4D 26 C6 EE 27 EC DD A6 81 220	2C 2A 3B 2B 4A 51 2B 51 4B 51 4B 51 4B 51 51 51 51 51 51 51 51 51 51 51 51 51	6D	22E	* SEARCH * 1ST OCC LBØB9	LDX STX LDX STX JSR BEQ JSR BRA FROM AE URENCE STX JSR LEAX TSTA BNE LDB LDU BEQ LDD STD LDD CMPA BRA BRA	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB04E JDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1, X LBØCD #2*3 ,X++ LB10A ,X++ LB10A ,X++ DATTXT ,X #\$86 LBØB9	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER SEARCH FOR END OF CURRENT LINE OR SUBLINE MOVE X ONE PAST END OF LINE GHECK FOR END OF LINE BRANCH IF END OF SUBLINE OUT OF DATA ERROR GET NEXT 2 CHARACTERS OD ERROR IF END OF PROGRAM GET BASIC LINE NUMBER AND SAVE IT IN DATIXT GET AN INPUT CHARACTER DATA TOKEN? NO KEEP LOOKING
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2393 2394	BØAØ BØAA2 BØAA4 BØAAC BØAAC BØBØ BØBB2 BØBB4 BØBB7 BØBB BØBB BØBC BØCC1 BØCC3 BØCC3 BØCC7 BØCC9 BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP	27 81 10 9E 9F 9F 9D 27 BD 20 4D 26 C6 EE 27 DD A6 81 26 20	2C 2A 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B	6D	22E	* SEARCH * 1ST OCC LBØB9 LBØCD	LDX STX LDX STX JSR BEQ JSR BRA FROM AC URENCE STX JSR LEAX TSTA BNE LDB BEQ LDD STD LDD STD LDA CMPA BNE BRA AD AND	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D LB04E JDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1,X LBØCD #2*3 ,X++ LB10A ,X++ DATTXT ,X #\$86 LB099 INPUT COMMANDS	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATIXT  GET AN INPUT CHARACTER  DATA TOKEN?  NO KEEP LOOKING  YES
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2393	BØAØ BØA2 BØAA BØAA BØAAE BØBØ BØB2 BØB4 BØB7 BØB8 BØB6 BØC1 BØC3 BØC5 BØC7 BØC9 BØC6 BØC7 BØC7 BØC9 BØC8 BØC8 BØC8 BØC8 BØC8 BØC8 BØC8 BØC8	27 81 10 9E 9F 9D 27 BD 20 20 9F BD 26 C6 EE 27 EC DD A6 81 26 20 9E	2C 2A 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B	6D		* SEARCH * 1ST OCC LBØB9 LBØCD	LDX STX LDX STX JSR BEQ JSR BRA FORM AGE FORM AGE FORM AGE LEAX TSTA BNE LDD LDA LDA LDD BRE BRA BNE LDD LDA LDA LDA LDA LDA LDA LDA LDA LDA	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D CHARAD OF THE TOKEN FOR DATA CHARAD LAEE8 1,X  LBØCD #2*3 ,X++ LB10A ,X++ DATTXT ,X #\$86 LBØB9 LBØ69 LBØ69 LBØCD LBØCD LBØCD LBØCD LBØCD LBOCO LBO	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATTXT  GET AN INPUT CHARACTER  DATA TOKEM?  NO KEEP LOOKING  YES  GET DATA POINTER
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2389 2390 2391 2392 2393 2393 2394 2395	BØAØ BØA2 BØAA BØAA BØAE BØBØ BØB2 BØB4 BØB7 BØB9 BØBB BØC0 BØC1 BØC5 BØC7 BØC5 BØC7 BØC5 BØC7 BØC5 BØC7 BØC5 BØC7 BØC5 BØC7 BØC5 BØC7 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5 BØC5	27 81 10 9E 9F 9F 9D 27 8D 20 9F 8D 30 4D 26 C6 EE 27 EC DD A6 81 26 20 9E 9E 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F	2C 2C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C	6D		* SEARCH * 1ST OCC LBØB9 LBØCD	LDX STX LDX STX JSR BEQ JSR BRA FORM AGE FORM AGE FORM AGE LEAX TSTA BNE LDD LDA LDA LDD BRE BRA BNE LDD LDA LDA LDA LDA LDA LDA LDA LDA LDA	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D CHESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1, X  LBØCD #2*3 ,X++ LB1ØA ,X++ LB1ØA ,X++ DATTXT ,X #\$86 LBØB9 LBØ69 INPUT COMMANDS DATTMP INPFLG	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER SEARCH FOR END OF CURRENT LINE OR SUBLINE MOVE X ONE PAST END OF LINE CHECK FOR END OF SUBLINE OUT OF DATA ERROR GET NEXT 2 CHARACTERS OD ERROR IF END OF PROGRAM GET BASIC LINE NUMBER AND SAVE IT IN DATTXT GET AN INPUT CHARACTER DATA TOKEN? NO KEEP LOOKING YES  GET DATA POINTER  * CHECK INPUT FLAG
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2391 2392 2393 2394 2393 2394 2395	BØAØ BØA2 BØAA BØAA BØAAE BØBØ BØBB BØBB BØBB BØB	27 81 10 9E 9F 9E 9D 27 8D 20 9F 8D 26 6C 6E 81 26 27 8D 4D 26 8D 4D 26 8D 8D 8D 8D 8D 8D 8D 8D 8D 8D 8D 8D 8D	2C 2C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C	6D		* SEARCH * 1ST OCC LBØB9 LBØCD	LDX STX LDX STX JSR BEQ JSR BRA BRA STX BRA BRA LEAX STX LEAX BNE LDB LDB LDD LDA CMPA BNE BNE LDA	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D LB04E JDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEE8 1,X LBØCD #2*3 ,X++ LB10A ,X++ DATTXT ,X #\$86 LB099 INPUT COMMANDS DATTMP INPFLG LADE8	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATTXT  GET AN INPUT CHARACTER  DATA TOKEN?  NO KEEP LOOKING  YES  GET DATA POINTER  * CHECK IMPUT FLAG  * SAVE NEW DATA POINTER IF READ
2366 2367 2368 2379 2371 2372 2373 2374 2375 2376 2377 2378 2379 2388 2381 2382 2383 2384 2385 2386 2387 2388 2389 2391 2392 2393 2391 2392 2393 2394 2395 2396 2397 2398	BØAØ BØA2 BØAA BØAA BØAAE BØBØ BØB2 BØBB BØBB BØBB BØBB BØC1 BØC3 BØC7 BØC9 BØC7 BØC9 BØCB BØCD BØCD BØCD BØCD BØCD BØCD BØCD	27 81 10 9E 9F 9F 9D 27 8D 20 30 4D 26 6 81 26 81 26 9E 9F 8D 30 4D 6 6 6 8 1 8 1 8 1 8 1 8 1 8 1 8 1 8 1 8	2C 2A6 32B AA5 2B95 AAE1 ØØ81 481 3886694 3996846	6D E8		* SEARCH * 1ST OCC LBØB9 LBØCD	LDX STX LDX STX LDX STX JSR BEQ BRA BRA BRA BRA LEAX LDB LDD LDD LDD LDA CMPA BRE BRA BRE BRA LDD LDA LDA LDA LDA LDA LDA LDB LDA LDB LDA LDB LDA LDB	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D LB04E DDRESS IN X FOR OF THE TOKEN FOR DATA CHARAD LAEEB 1,X LBØCD #2*3 ,X++ LB10A ,X++ DATTXT ,X #\$86 LB099 LB069 LB069 LB069 INPUT COMMANDS DATTMP INPFLG LADEB ,X	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET INPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATIXT  GET AN INPUT CHARACTER  DATA TOKEN?  NO KEEP LOOKING  YES  GET DATA POINTER  * CHECK INPUT FLAG  * SAVE NEW DATA POINTER IF READ  = CHECK NEXT CHARACTER IN INPUT BUFFER
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2380 2381 2382 2383 2384 2385 2386 2387 2389 2391 2392 2393 2391 2392 2393 2394 2395 2396 2397 2396	BØAØ BØAA2 BØAAA BØAAE BØABE BØBB BØBBB BØBBB BØBBBBBBBBBB	27 81 10 9E 9F 9F 27 8D 20 9F 8D 26 C6 EE 27 EC DD A6 81 26 20 9E	2C 2A6 32B 4A5 2B95 AAE1 ØØ6 4B1 3B96 8B96 8B96 8B96 8B96 8B96 8B96 8B96 8	6D E8		* SEARCH * 1ST OCC LBØB9 LBØCD	LDX STX LDX STX JSR BEQ JSR BRA FORM AGE FORM AGE LEAX TSTA BNE LDU BEQ LDD LDA LDA LDA BNE BNE LDD LDA LDA LDA LDA LDA LDB LDA LDA LDB LDA LDB LBNE LDB LDB LBNE LDB LDB LBNE LDB LDB LBNE LDB	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D CHARAD OF THE TOKEN FOR DATA CHARAD LAEE8 1,X  LBØCD #2*3 ,X++ LB1ØA ,X++ DATTXT ,X #\$86 LBØB9 LBØ69 INPUT COMMANDS DATTMP INPFLG LADE8 ,X LBØE7	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT GET CURRENT CHARACTER FROM BASIC BRANCH IF END OF LINE - EXIT COMMAND SYNTAX CHECK FOR COMMA GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER SEARCH FOR END OF CURRENT LINE OR SUBLINE MOVE X ONE PAST END OF LINE CHECK FOR END OF SUBLINE OUT OF DATA ERROR GET NEXT 2 CHARACTERS OD ERROR IF END OF PROGRAM GET BASIC LINE NUMBER AND SAVE IT IN DATTXT GET AN INPUT CHARACTER DATA TOKEN? NO KEEP LOOKING YES  GET DATA POINTER  * CHECK INPUT FLAG  * SAVE NEW DATA POINTER IF READ  = CHECK NEXT CHARACTER IN INPUT BUFFER  = RETURN IF NO MORE DATA FOR INPUT
2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2397 2398	BØAØ BØAA2 BØAAA BØAAC BØABB BØBBB BØBBBBBBBBBBBBB	27 81 10 9E 9F 9D 27 8D 20 9F 8D 26 C6 EE 27 EC DD A6 10 40 26 27 EC DD 40 40 40 40 40 40 40 40 40 40 40 40 40	2C 2A6 32B 4A5 2B95 AAE1 ØØ6 4B1 3B96 8B96 8B96 8B96 8B96 8B96 8B96 8B96 8	6D E8		* SEARCH * 1ST OCC LBØB9 LBØCD	LDX STX LDX STX JSR BEQ JSR BRA BRA STX STX JSR BRA LEAX STX LEAX BNE LDB LDD LDD LDD LDD LDD LDD LDD LDD LDD	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D LB04E OF THE TOKEN FOR DATA CHARAD LAEE8 1,X  LBØCD #2*3 ,X++ LB1ØA ,X++ DATTXT ,X #\$86 LB089 LB069 INPUT COMMANDS DATTMP INPFLG LADE8 ,X LB0E7 #LB0E7 #LB0E8-1	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATTXT  GET AN INPUT CHARACTER  DATA TOKEN?  NO KEEP LOOKING  YES  GET DATA POINTER  * CHECK IMPUT FLAG  * SAVE NEW DATA POINTER IF READ  = CHECK NEXT CHARACTER IN INPUT BUFFER  = RETURN IF NO MORE DATA FOR INPUT  POINT X TO ? EXTRA IGNORED
2366 2367 2368 2379 2371 2372 2373 2374 2375 2376 2377 2378 2378 2381 2382 2383 2383 2384 2385 2386 2387 2391 2392 2393 2391 2392 2393 2394 2395 2397 2398 2397 2398 2399 2409 2401	BØAØ BØAA2 BØAAA BØAAC BØABBØB BØBBB BØBBP BØBBB BØBBB BØBC BØCC1 BØCC3 BØCC5 BØCC7 BØCC9 BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP BØCCP	27 81 10 9E 9F 9D 27 8D 20 9F 8D 26 C6 EE 27 EC DD A6 10 40 26 27 EC DD 40 40 40 40 40 40 40 40 40 40 40 40 40	2C 2A6 32B 4A5 2B95 AAE1 ØØ6 4B1 3B96 8B96 8B96 8B96 8B96 8B96 8B96 8B96 8	6D E8		* SEARCH * 1ST OCC LBØB9 LBØCD * EXIT RE LBØD5	LDX STX LDX STX JSR BEQ JSR BRA BRA BRA BRA LOB LDB LDB LDB LDB LDB LDB LDB LDB LDB LD	DATTMP BINVAL CHARAD GETCCH LBØD5 LB26D LB26D LB26D LB04E OF THE TOKEN FOR DATA CHARAD LAEE8 1,X  LBØCD #2*3 ,X++ LB1ØA ,X++ DATTXT ,X #\$86 LB089 LB069 INPUT COMMANDS DATTMP INPFLG LADE8 ,X LB0E7 #LB0E7 #LB0E8-1	* GET CURRENT INPUT  * POINTER (USED AS A DATA POINTER) AND SAVE IT  * RESET IMPUT POINTER TO INPUT OR  * READ STATEMENT  GET CURRENT CHARACTER FROM BASIC  BRANCH IF END OF LINE - EXIT COMMAND  SYNTAX CHECK FOR COMMA  GET ANOTHER INPUT OR READ ITEM  RESET BASIC S INPUT POINTER  SEARCH FOR END OF CURRENT LINE OR SUBLINE  MOVE X ONE PAST END OF LINE  CHECK FOR END OF LINE  BRANCH IF END OF SUBLINE  OUT OF DATA ERROR  GET NEXT 2 CHARACTERS  OD ERROR IF END OF PROGRAM  GET BASIC LINE NUMBER AND  SAVE IT IN DATTXT  GET AN INPUT CHARACTER  DATA TOKEN?  NO KEEP LOOKING  YES  GET DATA POINTER  * CHECK IMPUT FLAG  * SAVE NEW DATA POINTER IF READ  = CHECK NEXT CHARACTER IN INPUT BUFFER  = RETURN IF NO MORE DATA FOR INPUT  POINT X TO ? EXTRA IGNORED

2404	BØE8	3F 45 58 54 !	52 41	LBØE8	FCC	'?EXTRA	IGNORED'	?EXTRA IGNORED MESSAGE
2405		20 49 47 4E						
2406	BØF4	45 44						
24Ø7 24Ø8	BØF6	ØD ØØ			FCB	CR,\$00		
2409				* NEXT				
2410	BØF8	26 Ø4		NEXT	BNE	LBØFE		BRANCH IF ARGUMENT GIVEN
2411		9E 8A			LDX	ZERO		X = Ø: DEFAULT FOR NO ARGUMENT
2412 2413		20 03		LDAEE	BRA JSR	LB1Ø1 LB357		EVALUATE AN ALDUA EVDDESSION
2413		BD B3 57 9F 3B		LBØFE LB1Ø1	STX	VARDES		EVALUATE AN ALPHA EXPRESSION SAVE VARIABLE DESCRIPTOR POINTER
2415		BD AB F9			JSR	LABF9		GO SCAN FOR FOR/NEXT DATA ON STACK
2416	B1Ø6	27 Ø4			BEQ	LB1ØC		BRANCH IF DATA FOUND
2417		C6 ØØ			LDB	#Ø		NEXT WITHOUT FOR ERROR (SHOULD BE CLRB)
2418 2419		20 47		LB1ØA	BRA	LB153		PROCESS ERROR POINT S TO START OF FOR/NEXT DATA
2419		1F 14 3Ø Ø3		LB1ØC	TFR LEAX	X,S 3 X		POINT X TO FP VALUE OF STEP
2421		BD BC 14			JSR	LBC14		COPY A FP NUMBER FROM (X) TO FPAØ
2422		A6 68			LDA	8,5		GET THE DIRECTION OF STEP
2423		97 54			STA	FPØSGN		SAVE IT AS THE SIGN OF FPAØ
2424 2425	B117	9E 3B BD B9 C2			LDX JSR	VARDES LB9C2		POINT (X) TO INDEX VARIABLE DESCRIPTOR ADD (X) TO FPAØ (STEP TO INDEX)
2426		BD BC 33			JSR	LBC33		PACK FPAØ AND STORE IT IN ADDRESS
2427				*				CONTAINED IN VARDES
2428	B11F				LEAX	9,5		POINT (X) TO TERMINAL VALUE OF INDEX
2429		BD BC 96			JSR	LBC96		COMPARE CURRENT INDEX VALUE TO TERMINAL VALUE OF INDEX
243Ø 2431	B124	EØ 68		*	SUBB	8,3		ACCB = Ø IF TERMINAL VALUE=CURRENT VALUE AND STEP=Ø OR IF STEP IS POSITIVE AND CURRENT VALUE>TERMINAL VALUE OR
2432				*				STEP IS NEGATIVE AND CURRENT VALUE <terminal td="" value<=""></terminal>
2433	B126	27 ØC			BEQ	LB134		BRANCH IF FOR/NEXT LOOP DONE
2434		AE 6E			LDX	14,5		* GET LINE NUMBER AND
2435		9F 68			STX	CURLIN		* BASIC POINTER OF
2436 2437		AE E8 10 9F A6			LDX STX	16,S CHARAD		* STATEMENT FOLLOWING THE * PROPER FOR STATEMENT
2438		7E AD 9E		LB131	JMP	LAD9E		JUMP BACK TO COMMAND INTEPR. LOOP
2439	B134	32 E8 12		LB134	LEAS	18,5		PULL THE FOR-NEXT DATA OFF THE STACK
2440		9D A5			JSR	GETCCH		GET CURRENT INPUT CHARACTER
2441 2442		81 2C 26 F4			CMPA BNE	#', LB131		CHECK FOR ANOTHER ARGUMENT RETURN IF NONE
2442	B13D	9D 9F			JSR	GETNCH		GET NEXT CHARACTER FROM BASIC
2444	B13F				BSR	LBØFE		BSR SIMULATES A CALL TO NEXT FROM COMMAND LOOP
2445								
2446				* EVALUAT			RESSION	
2447 2448	B141 B143	8D 13 1C FE		LB141 LB143	BSR ANDCC	LB156		EVALUATE EXPRESSION AND DO A TYPE CHECK FOR NUMERIC CLEAR CARRY FLAG
2449		7D		LB145	FCB	##FE \$7D		OP CODE OF TST \$1A01 - SKIP TWO BYTES (DO
2450				*				NOT CHANGE CARRY FLAG)
2451	B146	1A Ø1		LB146	ORCC	#1		SET CARRY
2452				+ CTDING	TVDE M	DE CHECK	TE ENTERED AT 1	DIAC THEN WALTED DIVIC TO LITHE EDDOD
2453 2454								LB146 THEN VALTYP PLUS IS 'TM' ERROR LB143 THEN VALTYP MINUS IS 'TM' ERROR
2455							TYPE CHECK IS DON	
2456							SET, THEN 'TM' ERF	
2457		~~ ~~					CLEAR, THEN 'TM' E	
2458 2459	B148	ØD Ø6 25 Ø3		LB148	TST BCS	VALTYP LB14F		TEST TYPE FLAG; DO NOT CHANGE CARRY BRANCH IF STRING
2460		2A 99			BPL	LBØE7		RETURN ON PLUS
2461	B14E	80			FCB	SKP2		SKIP 2 BYTES - TM ERROR
2462		2B 96		LB14F	BMI	LBØE7		RETURN ON MINUS
2463 2464		C6 18 7E AC 46		LB153	LDB JMP	#12*2 LAC46		TYPE M1SMATCH ERROR PROCESS ERROR
2465	D133	7L AC 40		* EVALUAT				PROCESS ERROR
2466		8D 6E		LB156		LB1C6		BACK UP INPUT POINTER
2467	B158				CLRA			END OF OPERATION PRECEDENCE FLAG
	B159			I D 1 E A	FCB PSHS	SKP2		SKIP TWO BYTES SAVE FLAG (RELATIONAL OPERATOR FLAG)
2469 2470		34 Ø4 34 Ø2		LB15A	PSHS			SAVE FLAG (RELATIONAL OPERATOR FLAG) SAVE FLAG (PRECEDENCE FLAG)
2471		C6 Ø1			LDB			*
2472		BD AC 33				LAC33		* SEE IF ROOM IN FREE RAM FOR (B) WORDS
2473		BD B2 23				LB223		GO EVALUATE AN EXPRESSION
2474 2475		ØF 3F 9D A5		LB168		TRELFL GETCCH		RESET RELATIONAL OPERATOR FLAG GET CURRENT INPUT CHARACTER
2475	B100	JD AS		* CHECK F			PERATORS	GET CORRENT INFOT CHARACTER
2477	B16A	8Ø B2		LB16A	SUBA			TOKEN FOR >
		25 13				LB181		BRANCH IF LESS THAN RELATIONAL OPERATORS
2479		81 Ø3			CMPA			* DDANGU IE ODEATED THAN DELATIONAL ODEDITION
248Ø 2481		24 ØF 81 Ø1			BCC CMPA	LB181 #1		* BRANCH IF GREATER THAN RELATIONAL OPERATORS SET CARRY IF >
2482	B174				ROLA			CARRY TO BIT Ø
2483	B175	98 3F			EORA	TRELFL		* CARRY SET IF
2484		91 3F				TRELFL		* TRELFL = ACCA
		25 64				LB1DF		BRANCH IF SYNTAX ERROR : == << OR >>
2486 2487		97 3F 9D 9F				TRELFL GETNCH		BIT 0: >, BIT 1 =, BIT 2: < SAVE DESIRED RELATIONAL COMPARISON GET AN INPUT CHARACTER
2488		2Ø E9				LB16A		CHECK FOR ANOTHER RELATIONAL OPERATOR
2489				*				
2490		D6 3F		LB181		TRELFL		GET RELATIONAL OPERATOR FLAG
2491 2492		26 33 10 24 00 6B				LB1B8 LB1F4		BRANCH IF RELATIONAL COMPARISON BRANCH IF > RELATIONAL OPERATOR
2.72	5100				2200	20217		The state of the s

2493	B189	8B Ø7	,			ADDA	#7	SEVEN ARITHMETIC/LOGICAL OPERATORS
2494	B18B	24 67	,			BCC	LB1F4	BRANCH IF NOT ARITHMETIC/LOGICAL OPERATOR
2495		99 Ø6					VALTYP	ADD CARRY, NUMERIC FLAG AND MODIFIED TOKEN NUMBER
2496	B18F	10 27	Ø4	7 C		LBEQ	LB6ØF	BRANCH IF VALTYP = FF, AND ACCA = + TOKEN -
2497 2498	R193	89 FF				ADCA	#-1	CONCATENATE TWO STRINGS RESTORE ARITHMETIC/LOGICAL OPERATOR NUMBER
	B195					PSHS		* STORE OPERATOR NUMBER ON STACK; MULTIPLY IT BY 2
2500		48				ASLA		* THEN ADD THE STORED STACK DATA = MULTIPLY
2501		AB E				ADDA	*	* X 3; 3 BYTE/TABLE ENTRY
2502		8E AA				LDX	#LAA51	JUMP TABLE FOR ARITHMETIC & LOGICAL OPERATORS
25Ø3 25Ø4	B190	30 86			LB19F	LEAX PULS		POINT X TO PROPER TABLE GET PRECEDENCE FLAG FROM STACK
2505		A1 84			LDIJI	CMPA		COMPARE TO CURRENT OPERATOR
2506	B1A3	24 55				BCC	LB1FA	BRANCH IF STACK OPERATOR > CURRENT OPERATOR
2507	B1A5	8D 90	;			BSR	LB143	TM ERROR IF VARIABLE TYPE = STRING
25Ø8 25Ø9					+ ODEDAT:		NO DROCESSED TO OF HIGHER	PRECEDENCE THAN THE PREVIOUS OPERATION.
	B1A7	34 02			LB1A7	PSHS		SAVE PRECEDENCE FLAG
2511		8D 29				BSR	LB1D4	PUSH OPERATOR ROUTINE ADDRESS AND FPAØ ONTO STACK
	B1AB	9E 3E	)			LDX	RELPTR	GET POINTER TO ARITHMETIC/LOGICAL TABLE ENTRY FOR
2513					*			LAST CALCULATED OPERATION
2514 2515		35 Ø2 26 10				PULS BNE	A LB1CE	GET PRECEDENCE FLAG OF PREVIOUS OPERATION BRANCH IF NOT END OF OPERATION
	B1B1		'			TSTA	LBICE	CHECK TYPE OF PRECEDENCE FLAG
	B1B2		ØØ	6A			LB220	BRANCH IF END OF EXPRESSION OR SUB-EXPRESSION
2518	B1B6	2Ø 4E				BRA	LB2Ø3	EVALUATE AN OPERATION
2519	0100	ao a					AL COMPARISON HERE VALTYP	DIT 7 OF TYPE FLAG TO CARRY
2520 2521	B1B8 B1BA		'		LB1B8	ASL ROLB	VALITP	BIT 7 OF TYPE FLAG TO CARRY SHIFT RELATIONAL FLAG LEFT - VALTYP TO BIT Ø
	B1BB					BSR	LB1C6	MOVE THE INPUT POINTER BACK ONE
2523	B1BD	8E B1	СВ			LDX	#LB1CB	POINT X TO RELATIONAL COMPARISON JUMP TABLE
2524		D7 3F				STB	TRELFL	SAVE RELATIONAL COMPARISON DATA
2525		ØF Ø6				CLR	VALTYP LB19F	SET VARIABLE TYPE TO NUMERIC PERFORM OPERATION OR SAVE ON STACK
2526 2527	B1C4	20 D9				BRA	LB19F	PERFORM OPERATION OR SAVE ON STACK
2528	B1C6	9E A6	,		LB1C6	LDX	CHARAD	* GET BASIC S INPUT POINTER AND
2529	B1C8	7E AE	ВВ			JMP	LAEBB	* MOVE IT BACK ONE
2530							MPARISON JUMP TABLE	251.4520441. 2010.102041. 51.40
2531 2532	B1CB B1CC	64 B2 F4			LB1CB LB1CC	FCB FDB	\$64 LB2F4	RELATIONAL COMPARISON FLAG JUMP ADDRESS
2532	DICC	DZ 19			LBICC	100	LB214	OUNT ADDRESS
2534	B1CE	A1 84			LB1CE	CMPA	, Х	COMPARE PRECEDENCE OF LAST DONE OPERATION TO
2535					*			NEXT TO BE DONE OPERATION
2536		24 31				BCC	LB2Ø3	EVALUATE OPERATION IF LOWER PRECEDENCE
2537 2538	8102	2Ø D3	i			BRA	LB1A7	PUSH OPERATION DATA ON STACK IF HIGHER PRECEDENCE
2539					* PUSH OF	PERATOR	EVALUATION ADDRESS AND FP	AØ ONTO STACK AND EVALUATE ANOTHER EXPR
2540	B1D4	EC Ø1			LB1D4	LDD	1,X	GET ADDRESS OF OPERATOR ROUTINE
2541								
		34 Ø6				PSHS	,	SAVE IT ON THE STACK
2542	B1D8	8D Ø8				BSR	LB1E2	PUSH FPAØ ONTO STACK
2542 2543	B1D8 B1DA	8D Ø8 D6 3F	; ;			BSR LDB	LB1E2 TRELFL	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG
2542	B1D8 B1DA B1DC	8D Ø8	7B		LB1DF	BSR LDB	LB1E2	PUSH FPAØ ONTO STACK
2542 2543 2544	B1D8 B1DA B1DC	8D Ø8 D6 3F 16 FF	7B		* PUSH F	BSR LDB LBRA JMP PAØ ONT	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR
2542 2543 2544 2545 2546 2547	B1D8 B1DA B1DC	8D Ø8 D6 3F 16 FF	7B		* PUSH FF * 1-2,S =	BSR LDB LBRA JMP PAØ ONTO	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR IT ORDER MANTISSA
2542 2543 2544 2545 2546 2547 2548	B1D8 B1DA B1DC B1DF	8D Ø8 D6 3F 16 FF 7E B2	7B		* PUSH FI * 1-2,S = * 5,S = 5	BSR LDB LBRA JMP PAØ ONTO HIGH O	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ROER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR IT ORDER MANTISSA IN ACCA
2542 2543 2544 2545 2546 2547	B1D8 B1DA B1DC B1DF	8D Ø8 D6 3F 16 FF	7B		* PUSH FF * 1-2,S =	BSR LDB LBRA JMP PAØ ONTO	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR IT ORDER MANTISSA :IN ACCA GET SIGN OF FPAØ MANTISSA
2542 2543 2544 2545 2546 2547 2548 2549	B1D8 B1DA B1DC B1DF B1E2 B1E4	8D Ø8 D6 3F 16 FF 7E B2	7B		* PUSH FI * 1-2,S = * 5,S = 5	BSR LDB LBRA JMP PAØ ONTO =HIGH OF SIGN RI	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR IT ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y
2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8	BD Ø8 D6 3F 16 FF 7E B2 D6 54 A6 84 35 20 34 Ø4	7B 77		* PUSH FFF * 1-2,S = * 5,S = 5 LB1E2	BSR LDB LBRA JMP PAØ ONTO HIGH OF SIGN RI LDB LDA PULS PSHS	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK
2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA	BD Ø8 D6 3F 16 FF 7E B2 D6 54 A6 84 35 20 34 Ø4 D6 4F	7B 77		* PUSH FFF * 1-2,S = * 5,S = 5 LB1E2	BSR LDB LBRA JMP PAØ ONTO HIGH OF SIGN R LDB LDA PULS PSHS LDB	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR IT ORDER MANTISSA :IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK
2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC	BD Ø8 D6 3F 16 FF 7E B2 D6 54 A6 84 35 20 34 Ø4 D6 4F 9E 50	7B 777		* PUSH FFF * 1-2,S = * 5,S = 5 LB1E2	BSR LDB LBRA JMP PAØ ONTI =HIGH OI SIGN RI LDB LDA PULS PSHS LDB LDB	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPØB	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK
2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE	BD Ø8 D6 3F 16 FF 7E B2 D6 54 A6 84 35 20 34 Ø4 D6 4F 9E 50	7B 77		* PUSH FFF * 1-2,S = * 5,S = 5 LB1E2	BSR LDB LBRA JMP PAØ ONTI =HIGH OI SIGN R LDB LDA PULS PSHS LDB LDX LDU	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPØB	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR IT ORDER MANTISSA :IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK
2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE B1FØ	BD Ø8 D6 3F 16 FF 7E B2 D6 54 A6 84 35 20 34 Ø4 D6 4F 9E 50 DE 52	7B 77		* PUSH FFF * 1-2,S = * 5,S = 5 LB1E2	BSR LDB LBRA JMP PAØ ONTI HIGH OI SIGN R LDB LDA PULS PSHS LDB LDX LDB LDX LDB LDX LDB	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPØEXP FPAØ FPAØ FPAØ+2	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR IT ORDER MANTISSA :IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK
2542 2543 2544 2545 2546 2547 2548 2559 2551 2552 2553 2554 2555 2556 2557 2558	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE B1FØ	BD Ø8 D6 3F T6 FF TE B2 D6 54 A6 84 35 20 34 Ø4 D6 4F 9E 50 DE 52 34 54	7B 77		* PUSH FI * 1-2,S = * 5,S = \$ LB1E2 LB1E6 LB1EA	BSR LDB LBRA JMP PAØ ONTI SIGN RI LDB LDA PULS PSHS LDB LDB LDB LDB LDB LDB LDB LDB LDB LDB	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPPØ FPPØ FPAØ+2 U,X,B ,Y	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK * * JUMP TO ADDRESS IN Y
2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2557	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE B1FØ B1F2	BD Ø8 D6 3F 16 FF 7E B2 D6 54 A6 84 35 20 34 Ø4 D6 4F 9E 50 DE 52 34 54 6E A4	7B 77		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA	BSR LDB LBRA PAØ ONT'- HIGH OI SIGN R LDB LDA PULS PSHS LDB LDX LDU PSHS JMP	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FPA	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK * * * * * * * * * * * * * * * * * * *
2542 2543 2544 2545 2546 2547 2548 2550 2551 2552 2553 2554 2555 2556 2557 2559 2569	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE B1FØ B1F2	BD Ø8 D6 3F 16 FF 7E B2 D6 54 A6 84 35 20 34 Ø4 D6 4F 9E 50 DE 52 34 54 6E A4	7B 77		* PUSH FI * 1-2,S = * 5,S = \$ LB1E2 LB1E6 LB1EA	BSR LDB LBRA PAØ ONTI- HIGH OI SIGN RI LDB LDA PULS PSHS LDB LDX LDU PSHS JMP	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT ORDER MANTISSA: IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK * * JUMP TO ADDRESS IN Y  OUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO)
2542 2543 2544 2545 2546 2547 2548 2550 2551 2552 2553 2554 2555 2556 2557 2558 2556 2557 2560 2560 2560	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE B1FØ B1F2	BD Ø8 D6 3F TE B2 D6 54 A6 84 35 20 34 Ø4 D6 4F 9E 52 34 54 6E A4	7B 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA	BSR LDB LBRA JMP PAØ ONTI HIGH OI SIGN R LDB LDA PULS PSHS LDB LDV PSHS JMP HERE I LDX	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FPA	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK * * * * * * * * * * * * * * * * * * *
2542 2543 2544 2545 2546 2547 2548 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2560 2560 2561 2562 2563	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1E8 B1E7 B1F2 B1F4 B1F4 B1F4 B1F8 B1FA	8D Ø8 D6 3F T6 FF 7E B2 D6 54 A6 84 35 26 34 Ø4 D6 4F 9E 52 34 54 6E A4 9E 8A A6 E6 81 64	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA	BSR LDB LBRA JMP PAØ ONTI HIGH OI SIGN R LDB LDA PULS PSHS LDB LDX LDU PSHS JMP HERE I LDX LDU DSHS JMP	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK  * PUSH FPAØ ONTO THE STACK  *  *  UDUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IT END OF EXPRESSION  * CHECK FOR RELATIONAL COMPARISON FLAG
2542 2543 2544 2545 2546 2547 2548 2551 2552 2553 2554 2555 2555 2556 2566 2561 2562 2563 2564 2563 2564	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E6 B1E7 B1E7 B1F4 B1F6 B1F8 B1FA B1FA	BD Ø8 D6 3F TE B2 D6 54 A6 84 35 20 34 Ø4 D6 52 34 54 6E A4 9E 8A A6 E0 27 26 81 64 27 Ø3	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4	BSR LDBA JMP PAØ ONTI HIGH OI SIGN RI LDB LDA PULS LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDA PSHS LDB LDA PSHS LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDB LDB LDB LDB LDB LDB LDB LDB LDB	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN DOTHE STACK. ,S = EXPONEN ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPPEXP FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB22Ø #\$64 LB2Ø1	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  I ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK  *  *  *  *  *  *  *  *  *  *  *  *  *
2542 2543 2544 2545 2546 2547 2548 2559 2551 2552 2553 2555 2556 2557 2558 2558 2558 2559 2560 2561 2562 2563 2563 2564 2563	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE B1F4 B1F2 B1F4 B1F6 B1F8 B1FA B1FA B1FA	BD 68 64 66 A4 66 A4 66 A4 66 A4 67 66 A4 67 66 A4 67 66 A4 67 67 67 67 67 67 67 67 67 67 67 67 67	78 77		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4	BSR LDB LBR JMP PAØ ONTH HIGH OI SIGN RI LDB LDA PULS PSHS LDB LDX LDU PSHS JMP HERE I LDA BEQ CMPA BEQ CMPA BEQ	LB1E2 TRELFL LB15A LB277 D THE STACK. , S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET SIGN OF FPAØ MANTISSA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  UNUD - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING
2542 2543 2544 2545 2546 2547 2548 2559 2550 2552 2553 2555 2555 2556 2557 2558 2558 2559 2560 2561 2562 2563 2564 2565 2566 2565 2566 2565 2566 2566	B1D8 B1DA B1DF B1E2 B1E4 B1E8 B1E8 B1EA B1EE B1FØ B1F2 B1F4 B1F4 B1F5 B1F4 B1F6 B1F8 B1FA B1FC B1FC	BD 68 86 84 86 84 84 84 84 84 84 84 84 84 84 84 84 84	78 77		* PUSH FI * 1-2,S = * 5,S = 9 LB1E2  LB1E6  LB1EA  * BRANCH LB1F4  LB1FA	BSR LDBA JMP PAØ ONTI HIGH OI SIGN RI LDB LDA PULS LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDA PSHS LDB LDA PSHS LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDA LDB LDB LDB LDB LDB LDB LDB LDB LDB LDB	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB22Ø #\$64 LB201 LB143 RELPTR	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT ORDER MANTISSA IIN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  UDUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE
2542 2543 2544 2545 2546 2547 2548 2559 2551 2552 2553 2554 2555 2556 2556 2566 2561 2562 2563 2564 2565 2566 2564 2565 2566 2566 2566	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EC B1EE B1F4 B1F2 B1F4 B1F6 B1F8 B1FA B1FA B1FA	BD 68 31 16 FT 8 22 16 16 16 16 16 16 16 16 16 16 16 16 16	7B 77		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4	BSR LDB LBRA JMP PAØ ONTI- HIGH OISIGN R LDB LDA PSHS LDB LDV PSHS LDU PSHS JMP HERE I LDA BEQ CMPA BEQ JSR STX PULS	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB22Ø #\$64 LB201 LB143 RELPTR	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET SIGN OF STAGK GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  UNUD - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING
2542 2543 2544 2545 2546 2547 2548 2559 2550 2553 2555 2556 2557 2558 2558 2560 2560 2561 2562 2563 2564 2565 2566 2566 2566 2566 2566 2566	B1D8 B1DA B1DF B1E2 B1E4 B1E6 B1E8 B1EA B1EE B1FØ B1F2 B1F4 B1F6 B1F8 B1FA B1FA B1FA B1FA B1FC B1FE B2Ø1 B2Ø3 B2Ø3 B2Ø3	BD 654 A6 84 B6 84 B6 84 B6 84 B6 84 B7 86 B8 86	7B 77		* PUSH FI * 1-2,S = * 5,S = 9 LB1E2  LB1E6  LB1EA  * BRANCH LB1F4  LB1FA	BSR LDB LBRA JMP PAØ ONTI- =HIGH OI SIGN RI LDB LDA PULS PSHS LDB LDU PSHS JMP HERE I LDX LDU PSHS JMP HERE I LDX LDX LDX LDX LDX LDX LDX LDX LDX LDX	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RORE MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  *  JUMP TO ADDRESS IN Y  OUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE GET RELATIONAL OPERATOR FLAG FROM STACK CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON RETURN IF NOT - NO RELATIONAL COMPARISON
2542 2543 2544 2545 2546 2547 2548 2550 2551 2552 2553 2554 2555 2556 2557 2558 2560 2561 2562 2563 2564 2562 2563 2564 2566 2567 2566 2567 2566 2567 2568 2567 2568 2567 2568 2569	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E6 B1E6 B1F2 B1F4 B1F2 B1F4 B1F6 B1F8 B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FC B1FC B1FC B1FC B1FC B1FC B1FC	8D 68 31 16 FF 82 26 34 54 54 54 54 54 54 54 54 54 54 54 54 54	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = 9 LB1E2  LB1E6  LB1EA  * BRANCH LB1F4  LB1FA	BSR LDB LBRA JMP PAØ ONTI- HIGH OI SIGN RI LDB LDA PULS LDB LDX LDB LDX LDU PSHS JMP HERE I LDA BEQ JSR CMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ CMPA BEQ CMPA BEQ CMPA CMPA CMPA	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222 #\$7D	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IT ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  UOUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR GET RELATIONAL OPERATOR GET RELATIONAL OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG
2542 2543 2544 2545 2546 2547 2548 2551 2552 2553 2554 2555 2556 2556 2566 2566 2562 2563 2564 2565 2564 2565 2566 2567 2568 2567 2568 2569 2567 2568	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E6 B1E6 B1F2 B1F4 B1F2 B1F4 B1F6 B1F8 B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FC B1FC B1FC B1FC B1FC B1FC B1FC	BD 654 A6 84 B6 84 B6 84 B6 84 B6 84 B7 86 B8 86	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = 9 LB1E2  LB1E6  LB1EA  * BRANCH LB1F4  LB1FA	BSR LDB LBRA JMP PAØ ONTI- HIGH OI SIGN RI LDB LDA PULS LDB LDX LDB LDX LDU PSHS JMP HERE I LDA BEQ JSR CMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ JSR VMPA BEQ CMPA BEQ CMPA BEQ CMPA CMPA CMPA	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RORE MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  *  JUMP TO ADDRESS IN Y  OUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE GET RELATIONAL OPERATOR FLAG FROM STACK CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON RETURN IF NOT - NO RELATIONAL COMPARISON
2542 2543 2544 2545 2546 2547 2548 2550 2551 2552 2553 2554 2555 2556 2557 2558 2560 2561 2562 2563 2564 2562 2563 2564 2566 2567 2566 2567 2566 2567 2568 2567 2568 2567 2568 2569	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E6 B1E6 B1F2 B1F4 B1F2 B1F4 B1F6 B1F8 B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FC B1FC B1FC B1FC B1FC B1FC B1FC	8D 68 31 16 FF 82 26 34 54 54 54 54 54 54 54 54 54 54 54 54 54	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = 5 LB1E6 LB1EA  * BRANCH LB1F4  LB1FA  LB2Ø1 LB2Ø3	BSR LDB LBRA JMP PAØ ONTI- =HIGH OI SIGN RI LDB LDA PSHS LDB LDD LDU PSHS JMP HERE I LDX LDA PSHS JMP HERE I LDX LDA PSHS JMP HERE I LDX LDA PSHS JMP HERE I LDX LDA PSHS JMP HERE I LDX LDA PSHS LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDX LDB LDB LDX LDB LDB LDB LDB LDB LDB LDB LDB LDB LDB	LB1E2 TRELFL LB15A LB277 D THE STACK. , S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222 #\$70 LB222	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IT ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  UOUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR GET RELATIONAL OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG
2542 2543 2544 2545 2546 2547 2548 2559 2551 2552 2553 2555 2556 2557 2568 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2568 2568 2568 2568 2568 2568 2568	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E6 B1E6 B1F2 B1F4 B1F2 B1F4 B1F6 B1F8 B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FC B1FC B1FC B1FC B1FC B1FC B1FC	8D 68 31 16 FF 82 26 34 54 54 54 54 54 54 54 54 54 54 54 54 54	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = 9 LB1E2 LB1E6 LB1EA  * BRANCH LB1F4  LB1FA  LB2Ø1 LB2Ø3  * EVALUA	BSR LDB LBRA JMP PAØ ONTI- HIGH OI SIGN R LDB LDA PULS PSHS LDB LDV PSHS JMP HERE I LDX LDU PSHS JMP HERE I LDX LDU PSHS JMP HERE I LDX LDW PSHS JMP HERE I LDX LDW PSHS JMP HERE I LDX LDW PSHS JMP HERE I LDX LDW PSHS LDW PSHS JMP CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ B BEQ B BEQ B B B BEQ B B B B B B B	LB1E2 TRELFL LB15A LB277 D THE STACK. ,S = EXPONEN RDER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB22Ø #\$64 LB201 LB143 RELPTR B #\$5A LB222 #\$7D LB222	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  *  JUMP TO ADDRESS IN Y  OUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE GET RELATIONAL OPERATOR ROUTINE GET RELATIONAL OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UMARY) FLAG RETURN IF NOT - NO RELATIONAL COMPARISON
2542 2543 2544 2545 2546 2547 2548 2559 2551 2552 2553 2556 2556 2556 2556 2566 2561 2562 2563 2564 2565 2566 2567 2568 2568 2568 2568 2569 2571 2572 2573 2574 2573 2574 2575	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E6 B1E6 B1F2 B1F4 B1F2 B1F4 B1F6 B1F8 B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FE B1FA B1FC B1FC B1FC B1FC B1FC B1FC B1FC B1FC	8D 68 31 16 FF 82 26 34 54 54 54 54 54 54 54 54 54 54 54 54 54	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4 LB1FA LB2Ø1 LB2Ø3	BSR LDB LBRA JMP PAØ ONTI- EHIGH OI SIGN R LDB LDA PSHS LDB LDV PSHS JMP HERE I LDX BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ	LB1E2 TRELFL LB15A LB277 D THE STACK. , S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222 #\$7D LB222 #\$7D LB222 PERATION. EIGHT BYTES WILL PRY FLOATING POINT RESULT T THE OPERATION. THE RTS AT	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IT ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  *  UOUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE GET RELATIONAL OPERATOR FLAG FROM STACK CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION - NO RELATIONAL COMPARISON
2542 2543 2544 2545 2546 2547 2548 2559 2550 2553 2555 2556 2557 2558 2560 2561 2562 2563 2564 2565 2566 2567 2568 2566 2567 2568 2569 2568 2569 2569 2570 2570 2570 2570 2570 2570 2570 2570	B1D8 B1DA B1DA B1DF B1E2 B1E4 B1E8 B1EA B1EE B1FØ B1F2 B1F4 B1F6 B1F8 B1FA B1FA B1FA B1FC B2Ø3 B2Ø3 B2Ø3 B2Ø9 B2Ø9	8D 68 68 68 68 68 68 68 68 68 68 68 68 68	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4 LB1FA LB2Ø1 LB2Ø3	BSR LDB JMP PAØ ONTI= HIGH OI SIGN R LDB LDA PULS PSHS LDB LDV PSHS LDU PSHS JMP HERE I LDX LDQ CMPA BEQ CMPA CMPA BEQ CMPA CMPA CMPA CMPA CMPA CMPA CMPA CMPA	LB1E2 TRELFL LB15A LB277 D THE STACK. , S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPPØEXP FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222 PERATION. EIGHT BYTES WILL RY FLOATING POINT RESULT T	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  *  *  JUMP TO ADDRESS IN Y  OUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE GET RELATIONAL OPERATOR FLAG FROM STACK CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION ON RELATIONAL COMPARISON  BE STORED ON STACK, FIRST SIX BYTES HEN THE ADDRESS OF ROUTINE WHICH 'END OF ROUTINE WILL VECTOR
2542 2543 2544 2545 2546 2547 2548 2559 2551 2552 2553 2554 2555 2556 2560 2561 2562 2563 2564 2565 2566 2567 2568 2567 2568 2569 2571 2572 2573 2574 2575 2576	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1E8 B1F2 B1F4 B1F2 B1F4 B1F6 B1F8 B1F8 B1FC B1FE B2Ø1 B2Ø3 B2Ø5 B2Ø7 B2Ø8	8D 66 31 16 82 16 82 16 84 16 84	78 77		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4 LB1FA LB2Ø1 LB2Ø3	BSR LDB LBRA JMP PAØ ONTI- FIGN R: LDB LDA LDB LDA LDB LDX LDU PSHS LDV PSHS LDA LDA BEQ CMPA BEQ BEQ CMPA BEQ	LB1E2 TRELFL LB15A LB277 D THE STACK. , S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPPØEXP FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222 PERATION. EIGHT BYTES WILL RY FLOATING POINT RESULT T THE OPERATION. THE RTS AT ROUTINE.	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK  * PUSH FPAØ ONTO THE STACK  *  *  *  *  *  *  *  *  *  *  *  *  *
2542 2543 2544 2545 2546 2548 2549 2550 2552 2553 2556 2556 2557 2566 2561 2562 2563 2564 2565 2566 2567 2568 2569 2568 2569 2571 2572 2573 2574 2575 2577 2578	B1D8 B1DA B1DA B1DF B1E2 B1E4 B1E8 B1EA B1EE B1FØ B1F2 B1F4 B1F6 B1F8 B1FA B1FA B1FA B1FC B2Ø3 B2Ø3 B2Ø3 B2Ø9 B2Ø9	8D 68 31 17	787777777777777777777777777777777777777		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4 LB1FA LB2Ø1 LB2Ø3	BSR LDB LDBA JMP PAØ ONTI- SIGN R. LDB LDA PSHS LDB LDX LDU PSHS JMP HERE I LDA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ CMPA BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ	LB1E2 TRELFL LB15A LB277 D THE STACK. , S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPØEXP FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$64 LB201 LB143 RELPTR B #\$5A LB222 #\$7D LB222 #\$7D LB222 PERATION. EIGHT BYTES WILL PRY FLOATING POINT RESULT T THE OPERATION. THE RTS AT	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  *  *  JUMP TO ADDRESS IN Y  OUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE GET RELATIONAL OPERATOR FLAG FROM STACK CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION ON RELATIONAL COMPARISON  BE STORED ON STACK, FIRST SIX BYTES HEN THE ADDRESS OF ROUTINE WHICH 'END OF ROUTINE WILL VECTOR
2542 2543 2544 2545 2546 2547 2548 2559 2550 2553 2555 2556 2557 2558 2560 2561 2562 2563 2564 2563 2564 2565 2566 2567 2568 2568 2569 2570 2570 2570 2570 2570 2571 2572 2573 2574 2577 2577 2577 2577 2577 2577 2577	B1D8 B1DA B1DC B1DF B1E2 B1E4 B1E6 B1E8 B1E8 B1F2 B1F4 B1F6 B1F8 B1F7 B1F6 B1F8 B1F9 B1F9 B205 B205 B205 B207 B208 B208 B208 B208 B208 B208 B208 B208	8D 68 31 16 FF 82 64 65 84 65 84 76 85 85 85 85 85 85 85 85 85 85 85 85 85	78 77 77 77 77 77 77 77 77 77 77 77 77 7		* PUSH FI * 1-2,S = * 5,S = S LB1E2 LB1E6 LB1EA * BRANCH LB1F4 LB1FA LB2Ø1 LB2Ø3	BSR LDB LBRA JMP PAØ ONTI- HIGH OI HI HIGH OI HI	LB1E2 TRELFL LB15A LB277 D THE STACK. , S = EXPONEN ROBER MANTISSA 3-4,S = LOW ETURN WITH PRECEDENCE CODE FPØSGN ,X Y B FPPØEXP FPPØ FPAØ FPAØ+2 U,X,B ,Y F NON-OPERATOR CHARACTER F ZERO ,S+ LB220 #\$564 LB201 LB143 RELPTR B #\$5A LB222 #\$7D LB222 PERATION. EIGHT BYTES WILL RY FLOATING POINT RESULT T THE OPERATION. THE RTS AT ROUTINE. RELFLG	PUSH FPAØ ONTO STACK GET BACK RELATIONAL OPERATOR FLAG EVALUATE ANOTHER EXPRESSION SYNTAX ERROR  IT  ORDER MANTISSA IN ACCA GET SIGN OF FPAØ MANTISSA GET PRECEDENCE CODE TO ACCA GET SIGN OF FPAØ MANTISSA GET RETURN ADDRESS FROM STACK & PUT IT IN Y SAVE ACCB ON STACK * PUSH FPAØ ONTO THE STACK *  *  *  JUMP TO ADDRESS IN Y  OUND - USUALLY ) OR END OF LINE POINT X TO DUMMY VALUE (ZERO) GET PRECEDENCE FLAG FROM STACK BRANCH IF END OF EXPRESSION * CHECK FOR RELATIONAL COMPARISON TM ERROR IF VARIABLE TYPE = STRING SAVE POINTER TO OPERATOR ROUTINE GET RELATIONAL OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT OPERATOR RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION - NO RELATIONAL COMPARISON CHECK FOR NEGATION (UNARY) FLAG RETURN IF NEGATION COMPARISON CHECK FOR NOT ORDER RETURN IF NEGATION COMPARISON CHECK FOR NOT ORDER RETURN IF NEGATION COMPARISON CHECK FOR NOT ORDER RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT ORDER RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT ORDER RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT ORDER RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT ORDER RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT ORDER RETURN IF NOT - NO RELATIONAL COMPARISON CHECK FOR NOT ORDER RETURN IF NOT - NO RELATIONAL COMPARISON

2582	B216	DF 5F				STU	FPA1+2	*
2583	B218	35 Ø4				PULS	В	= GET MANTISSA SIGN AND
2584	B21A	D7 61				STB	FP1SGN	= SAVE IT IN FPA1
2585	B21C	D8 54				EORB	FPØSGN	EOR IT WITH FPA1 MANTISSA SIGN
2586	B21E	D7 62				STB	RESSGN	SAVE IT IN RESULT SIGN BYTE
2587	B22Ø	D6 4F			LB22Ø	LDB	FPØEXP	GET EXPONENT OF FPAØ
2588	B222	39			LB222	RTS		
2589								
2590	B223	BD Ø1	8B		LB223	JSR	RVEC15	HOOK INTO RAM
2591	B226	ØF Ø6				CLR	VALTYP	INITIALIZE TYPE FLAG TO NUMERIC
2592	B228	9D 9F				JSR	GETNCH	GET AN INPUT CHAR
2593	B22A	24 Ø3				BCC	LB22F	BRANCH IF NOT NUMERIC
2594	B22C	7E BD	12		LB22C	JMP	LBD12	CONVERT ASCII STRING TO FLOATING POINT -
2595					*			RETURN RESULT IN FPAØ
2596					* PROCESS	Δ ΝΩΝ	NUMERIC FIRST CH	
2597	B22F	BD B3	Δ2		LB22F	JSR	LB3A2	SET CARRY IF NOT ALPHA
2598	B232	24 50			LULL!	BCC	LB284	BRANCH IF ALPHA CHARACTER
2599	B234	81 2E				CMPA		IS IT . (DECIMAL POINT)?
2600	B236	27 F4				BEQ	LB22C	CONVERT ASCII STRING TO FLOATING POINT
2601	B238	81 AC				CMPA	#\$AC	MINUS TOKEN
2602		27 40				BEQ	LB27C	YES - GO PROCESS THE MINUS OPERATOR
2603	B23C	81 AB				CMPA	#\$AB	PLUS TOKEN
2604	B23E	27 E3				BEQ	LB223	YES - GET ANOTHER CHARACTER
2605	B24Ø	81 22				CMPA	#'"	STRING DELIMITER?
2606	B242	26 ØA				BNE	LB24E	NO
2607	B244	9E A6			LB244	LDX	CHARAD	CURRENT BASIC POINTER TO X
2608	B246	BD B5	18			JSR	LB518	SAVE STRING ON STRING STACK
2609	B249	9E 64			LB249	LDX	COEFPT	* GET ADDRESS OF END OF STRING AND
2610	B24B	9F A6				STX	CHARAD	* PUT BASIC S INPUT POINTER THERE
2611	B24D	39				RTS		
2612		81 A8			LB24E	CMPA	#\$A8	NOT TOKEN?
	B25Ø	26 ØD			LD24L	BNE	LB25F	NO NO
2613	6250	20 00			+ DDOOFEC			NU
2614					* PRUCESS		T OPERATOR	NOT - DESCEDENCE - 51.40
2615	B252	86 5A				LDA	#\$5A	NOT PRECEDENCE FLAG
2616	B254	BD B1				JSR	LB15A	PROCESS OPERATION FOLLOWING NOT
2617	B257	BD B3	ED			JSR	INTCNV	CONVERT FPAØ TO INTEGER IN ACCD
2618	B25A	43				COMA		* NOT THE INTEGER
2619	B25B	53				COMB		*
2620	B25C	7E B4	F4			JMP	GIVABF	CONVERT ACCD TO FLOATING POINT (FPAØ)
2621	B25F	4 C			LB25F	INCA		CHECK FOR TOKENS PRECEEDED BY 5FF
2622	B26Ø	27 2E				BEQ	LB29Ø	IT WAS PRECEEDED BY 5FF
2623	B262	8D Ø6			LB262	BSR	LB26A	SYNTAX CHECK FOR A (
2624	B264	BD B1	56			JSR	LB156	EVALUATE EXPRESSIONS WITHIN PARENTHESES AT
2625	5201	00 01	•		*	0011	25100	HIGHEST PRECEDENCE
2626	B267	C6 29			LB267	LDB	#')	SYNTAX CHECK FOR )
					LDZ07			
2627	B269	8C				FCB	SKP2	SKIP 2 BYTES
2628	B26A	C6 28			LB26A	LDB	#'(	SYNTAX CHECK FOR (
2629	B26C	8C				FCB	SKP2	SKIP 2 BYTES
2630	B26D	C6 2C			LB26D	LDB	#',	SYNTAX CHECK FOR COMMA
2631	B26F	E1 9F	ØØ	A6	LB26F	CMPB	[CHARAD]	* COMPARE ACCB TO CURRENT INPUT
2632	B273	26 Ø2				BNE	LB277	* CHARACTER - SYNTAX ERROR IF NO MATCH
2633	B275	ØE 9F				JMP	GETNCH	GET A CHARACTER FROM BASIC
2634	B277	C6 Ø2			LB277	LDB	#2*1	SYNTAX ERROR
2635	B279	7E AC	46			JMP	LAC46	JUMP TO ERROR HANDLER
2636								
2637					* PROCESS	THE MI	NUS (UNARY) OPER	ATOR
2638	B27C	86 7D			LB27C	LDA	#\$7D	MINUS (UNARY) PRECEDENCE FLAG
2639	B27E	BD B1	5 A		LDL70	JSR	LB15A	PROCESS OPERATION FOLLOWING UNARY NEGATION
2640	B281	7E BE				JMP	LBEE9	CHANGE SIGN OF FPAØ MANTISSA
	B281	/E BE	E9			JMP	LBEES	CHANGE SIGN OF FPAD MANIISSA
2641							=======================================	
2642							EXPRESSION	
2643		BD B3			LB284	JSR	LB357	FIND THE DESCRIPTOR ADDRESS OF A VARIABLE
2644		9F 52				STX	FPAØ+2	SAVE DESCRIPTOR ADDRESS IN FPAØ
2645	B289	96 Ø6				LDA	VALTYP	TEST VARIABLE TYPE
2646	B28B	26 95				BNE	LB222	RETURN IF STRING
2647	B28D	7E BC	14			JMP	LBC14	COPY A FP NUMBER FROM (X) TO FPAØ
2648								
2649					* EVALUAT	ING A S	ECONDARY TOKEN	
2650	B29Ø	9D 9F			LB29Ø	JSR	GETNCH	GET AN INPUT CHARACTER (SECONDARY TOKEN)
2651		1F 89				TFR	A,B	SAVE IT IN ACCB
2652	B294	58				ASLB	,-	X2 & BET RID OF BIT 7
2653		9D 9F				JSR	GETNCH	GET ANOTHER INPUT CHARACTER
2654		C1 26						19 SECONDARY FUNCTIONS IN BASIC
								BRANCH IF COLOR BASIC TOKEN
2655		23 Ø4		22		BLS	LB29F	JUMP TO EXBAS SECONDARY TOKEN HANDLER
2656		6E 9F		34	1 0205	JMP	[COMVEC+18]	
2657		34 Ø4			LB29F	PSHS		SAVE TOKEN OFFSET ON STACK
2658		C1 1C				CMPB	#2*14	CHECK FOR NUMERIC ARGUMENT TOKEN
2659		25 22				BCS	LB2C7	DO SECONDARIES \$8D (JOYSTK) OR LESS
2660		C1 24				CMPB		*
2661		24 20				BCC	LB2C9	* DO SECONDARIES \$92 (INKEY\$) OR >
		8D BF				BSR	LB26A	SYNTAX CHECK FOR A (
2662	B2A9					LDA	,\$	GET TOKEN NUMBER
2662 2663	B2A9 B2AB	A6 E4						CHECK FOR POINT COMMAND
	B2AB	A6 E4 81 22						
2663 2664	B2AB B2AD	81 22				BCC	LB2C9	DO POINT COMMAND (\$91)
2663 2664 2665	B2AB B2AD				* DO SECO	BCC	LB2C9 \$8F \$8F \$90 (	DO POINT COMMAND (\$91)
2663 2664 2665 2666	B2AB B2AD B2AF	81 22 24 18			* DO SECO	NDARIES	\$8E, \$8F, \$9Ø (	LEFT\$, RIGHT\$, MID\$)
2663 2664 2665 2666 2667	B2AB B2AD B2AF B2B1	81 22 24 18 BD B1			* DO SECO	NDARIES JSR	\$8E, \$8F, \$9Ø ( LB156	LEFT\$, RIGHT\$, MID\$) EVALUATE FIRST STRING IN ARGUMENT
2663 2664 2665 2666 2667 2668	B2AB B2AD B2AF B2B1 B2B4	81 22 24 18 BD B1 8D B7	56		* DO SECO	NDARIES JSR BSR	\$8E, \$8F, \$9Ø ( LB156 LB26D	LEFT\$, RIGHT\$, MID\$)  EVALUATE FIRST STRING IN ARGUMENT  SYNTAX CHECK FOR A COMMA
2663 2664 2665 2666 2667 2668 2669	B2AB B2AD B2AF B2B1 B2B4 B2B6	81 22 24 18 BD B1 8D B7 BD B1	56 46		* DO SECO	DARIES JSR BSR JSR	\$ \$8E, \$8F, \$9Ø ( LB156 LB26D LB146	LEFT\$, RIGHT\$, MID\$)  EVALUATE FIRST STRING IN ARGUMENT SYMTAX CHECK FOR A COMMA TM ERROR IF NUMERIC VARIABLE
2663 2664 2665 2666 2667 2668	B2AB B2AD B2AF B2B1 B2B4	81 22 24 18 BD B1 8D B7 BD B1	56 46		* DO SECO	NDARIES JSR BSR	\$ \$8E, \$8F, \$9Ø ( LB156 LB26D LB146	LEFT\$, RIGHT\$, MID\$)  EVALUATE FIRST STRING IN ARGUMENT  SYNTAX CHECK FOR A COMMA

2671	R2RR	DE 52			LDU	FPAØ+2	POINT U TO STRING DESCRIPTOR
2672	B2BD	34 42			PSHS		SAVE TOKEN OFFSET AND DESCRIPTOR ADDRESS
2673	B2BF	BD B7			JSR	LB7ØB	EVALUATE FIRST NUMERIC ARGUMENT
2674	B2C2	35 Ø2			PULS	A	GET TOKEN OFFSET FROM STACK
2675	B2C4	34 Ø6				В, А	SAVE TOKEN OFFSET AND NUMERIC ARGUMENT
2676 2677	B2C6 B2C7	8E 8D 99		10007	FCB BSR	\$8E	OP CODE OF LDX# - SKIP 2 BYTES
2678	B2C7	35 Ø4		LB2C7 LB2C9	PULS	LB262 B	SYNTAX CHECK FOR A ( GET TOKEN OFFSET
2679	B2CB	BE Ø1		LDLUJ	LDX	COMVEC+8	GET SECONDARY FUNCTION JUMP TABLE ADDRESS
2680	B2CE	3A		LB2CE	ABX		ADD IN COMMAND OFFSET
2681				*			
2682				* HERE IS		WE BRANCH TO A SECONDARY	
2683 2684	B2CF B2D1	AD 94 7E B1			JSR JMP	[,X] LB143	GO DO AN SECONDARY FUNCTION TM ERROR IF VARIABLE TYPE = STRING
2685	6201	/ E DI	43		UNIF	LB143	IN ERROR IF VARIABLE TIPE - STRING
2686				* LOGICAL	OPERA	TOR OR JUMPS HERE	
2687	B2D4	86		LB2D4	FCB	SKP1LD	SKIP ONE BYTE - OR FLAG = \$4F
2688							
2689	0005	45				TOR AND JUMPS HERE	AND FLAC
269Ø 2691	B2D5 B2D6	4F 97 Ø3		LB2D5	CLRA STA	TMPLOC	AND FLAG = Ø AND/OR FLAG
2692	B2D8	BD B3			JSR	INTCNV	CONVERT FPAØ INTO AN INTEGER IN ACCD
2693		DD Ø1			STD	CHARAC	TEMP SAVE ACCD
2694	B2DD	BD BC	4A		JSR	LBC4A	MOVE FPA1 TO FPAØ
2695	B2EØ	BD B3			JSR	INTCNV	CONVERT FPAØ INTO AN INTEGER IN ACCD
2696	B2E3	ØD Ø3 26 Ø6			TST	TMPLOC	CHECK AND/OR FLAG
2697 2698	B2E5 B2E7	26 Ø6			BNE	LB2ED CHARAC	BRANCH IF OR * AND ACCD WITH FPAØ INTEGER
2699	B2E9	D4 Ø2				ENDCHR	* STORED IN ENDCHR
2700	B2EB	20 04			BRA	LB2F1	CONVERT TO FP
2701	B2ED	9A Ø1		LB2ED	ORA	CHARAC	* OR ACCD WITH FPAØ INTEGER
2702	B2EF	DA Ø2			ORB	ENDCHR	* STORED IN CHARAC
27Ø3	B2F1	7E B4	F4	LB2F1	JMP	GIVABF	CONVERT THE VALUE IN ACCD INTO A FP NUMBER
2704				+ 051 4770		ADADICON DOCCECC HANDLED	
27Ø5 27Ø6	B2F4	BD B1	40	* RELATIO	JSR	MPARISON PROCESS HANDLER LB148	TM ERROR IF TYPE MISMATCH
2707	B2F7	26 10			BNE	LB309	BRANCH IF STRING VARIABLE
2708	B2F9	96 61			LDA	FP1SGN	* PACK THE MANTISSA
2709	B2FB	8A 7F			ORA	#\$7F	* SIGN OF FPA1 INTO
2710	B2FD	94 5D			ANDA	FPA1	* BIT 7 OF THE
2711	B2FF	97 5D			STA	FPA1	* MANTISSA MS BYTE
2712	B3Ø1	8E ØØ			LDX	#FP1EXP	POINT X TO FPA1
2713	B3Ø4	BD BC			JSR	LBC96	COMPARE FPAØ TO FPA1
2714 2715	B3Ø7	20 36			BRA	LB33F	CHECK TRUTH OF RELATIONAL COMPARISON
2715				* RELATIO	NAI COI	MPARISON OF STRINGS	
2717	B3Ø9	ØF Ø6		LB3Ø9	CLR	VALTYP	SET VARIABLE TYPE TO NUMERIC
2718	B3ØB	ØA 3F			DEC	TRELFL	REMOVE STRING TYPE FLAG (BITØ=1 FOR STRINGS) FROM THE
2719							
				*			DESIRED RELATIONAL COMPARISON DATA
2720	B3ØD	BD B6	57	*	JSR	LB657	GET LENGTH AND ADDRESS OF STRING WHOSE
2721				*			GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ
2721 2722	B31Ø	D7 56		*	STB	STRDES	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ * SAVE LENGTH AND ADDRESS IN TEMPORARY
2721 2722 2723	B310 B312	D7 56 9F 58		*	STB STX	STRDES STRDES+2	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ * SAVE LENGTH AND ADDRESS IN TEMPORARY * DESCRIPTOR (STRING B)
2721 2722	B31Ø	D7 56		*	STB	STRDES	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ * SAVE LENGTH AND ADDRESS IN TEMPORARY
2721 2722 2723 2724	B31Ø B312 B314	D7 56 9F 58 9E 5F BD B6	59	*	STB STX LDX	STRDES STRDES+2 FPA1+2	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ * SAVE LENGTH AND ADDRESS IN TEMPORARY * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING
2721 2722 2723 2724 2725 2726 2727	B310 B312 B314 B316 B319 B31B	D7 56 9F 58 9E 5F BD 86 96 56 34 Ø4	59	*	STB STX LDX JSR LDA PSHS	STRDES STRDES+2 FPA1+2 LB659 STRDES B	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK
2721 2722 2723 2724 2725 2726 2727 2728	B310 B312 B314 B316 B319 B31B B31D	D7 56 9F 58 9E 5F BD B6 96 56 34 Ø4 AØ EØ	59	*	STB STX LDX JSR LDA PSHS SUBA	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B
2721 2722 2723 2724 2725 2726 2727 2728 2729	B31Ø B312 B314 B316 B319 B31B B31D B31F	D7 56 9F 58 9E 5F BD B6 96 56 34 Ø4 AØ EØ 27 Ø7	59	*	STB STX LDX JSR LDA PSHS SUBA BEQ	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730	B310 B312 B314 B316 B319 B31B B31D B31F B321	D7 56 9F 58 9E 5F BD B6 96 56 34 Ø4 AØ EØ 27 Ø7 86 Ø1	59	*	STB STX LDX JSR LDA PSHS SUBA BEQ LDA	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731	B310 B312 B314 B316 B319 B31B B31D B31F B321 B323	D7 56 9F 58 9E 5F BD 86 96 56 34 Ø4 AØ EØ 27 Ø7 86 Ø1 24 Ø3	59	*	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730	B31Ø B312 B314 B316 B319 B31B B31D B31F B321 B323 B325	D7 56 9F 58 9E 5F BD B6 96 56 34 Ø4 AØ EØ 27 Ø7 86 Ø1	59	*	STB STX LDX JSR LDA PSHS SUBA BEQ LDA	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732	B310 B312 B314 B316 B319 B31B B31D B31F B321 B323 B325 B327	D7 56 9F 58 9E 5F BD 86 96 56 34 04 AØ EØ 27 07 86 01 24 03 D6 56	59	* * LB328	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735	B310 B312 B314 B316 B319 B31B B31D B31F B321 B323 B325 B327 B328 B32A	D7 56 9F 58 9E 5F BD 86 96 56 34 04 AØ EØ 27 07 86 01 24 03 D6 56 40 97 54 DE 58	59	*	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736	B310 B312 B314 B316 B319 B31B B31D B31F B321 B323 B325 B327 B328	D7 56 9F 58 9E 5F BD 86 96 56 34 04 AØ EØ 27 07 86 01 24 03 D6 56 40 97 54 DE 58	59	* LB328	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU INCB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737	B310 B312 B314 B316 B319 B31B B31D B31F B321 B323 B325 B327 B328 B328	D7 56 9F 58 9E 5F BD 86 96 56 34 04 AØ EØ 27 07 86 01 24 03 D6 56 40 97 54 DE 58 5C	59	* LB328 * ENTER W	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU INCB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH  TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B  SET FLAG = FALSE (1FF)  SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW  HORTER STRING
2721 2722 2723 2724 2725 2726 2727 2728 2730 2731 2732 2733 2734 2735 2736 2737	B310/ B312/ B314/ B316/ B319/ B31B/ B31D/ B31F/ B321/ B323/ B325/ B327/ B328/ B328/ B32C/ B32D/	D7 56 9F 58 9E 5F BD 86 96 56 34 04 A0 E0 27 07 86 01 24 03 D6 56 40 97 54 DE 58 5C	59	* LB328	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU INCB ITH ACC DECB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH
2721 2722 2723 2724 2725 2726 2727 2728 2730 2731 2732 2733 2734 2735 2736 2737 2738	B31Ø B312 B314 B316 B319 B31B B31D B31F B321 B323 B325 B327 B328 B32A B32C B32D	D7 56 9F 58 9E 5F BD 86 96 56 34 04 A0 E0 27 07 86 01 24 03 D6 56 40 97 54 DE 58 5C 5A 26 04	59	* LB328 * ENTER W	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU INCB ITH ACC DECB BNE	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING LENGTH BRANCH IF ALL OF STRING LENGTH
2721 2722 2723 2724 2725 2726 2727 2728 2730 2731 2732 2733 2734 2735 2736 2737	B31Ø B312 B314 B316 B319 B31B B31D B31F B321 B323 B325 B327 B328 B328 B32A B32C	D7 56 9F 58 9E 5F BD 86 96 56 34 04 A0 E0 27 07 86 01 24 03 D6 56 40 97 54 DE 58 5C	59	* LB328 * ENTER W	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU INCB ITH ACC DECB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740	B31Ø B312 B314 B316 B319 B31B B31D B31F B321 B323 B325 B327 B328 B32C B32C B32D B32E B33Ø B332	D7 56 9F 58 9E 5F BD 86 96 56 34 04 AØ EØ 27 07 86 01 24 03 D6 56 40 DF 58 5C SA 26 04 D6 54	59	* LB328 * ENTER W	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU INCB ITH ACC DECB BNE LDB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B  SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB  DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB
2721 2722 2723 2724 2725 2726 2727 2728 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2740 2740	B310 B312 B314 B316 B319 B31D B31F B321 B325 B327 B328 B32C B32C B32C B32C B32C B32C	D7 56 9F 58 9E 5F BD 86 96 56 34 04 AØ EØ 27 07 86 01 24 03 D6 56 40 97 54 DE 58 5C 5A 26 04 20 08 A6 80 A1 CØ	59	* LB328 * ENTER W LB32D	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU INCB ITH ACC DECB BNE LDB BRA LDB BRA LDA CMPA	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB337 ,X+ ,U+	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCA WITH LENGTH B  SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A COMPARE TO STRING B
2721 2722 2723 2724 2725 2726 2727 2728 2739 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744	B310 B312 B314 B316 B319 B31F B321 B323 B325 B327 B328 B32A B32C B32D B32E B330 B332 B334 B334 B334 B336	D7 56 9F 58 9E 5F BD 86 96 56 34 04 40 E0 27 07 86 01 24 03 D6 56 40 97 54 DE 58 5C  5A 26 04 26 08 A6 80 A6 80 A1 C0 27 F3	59	* LB328 * ENTER W LB32D	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB INCB LDU INCB LDU INCB LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE BNE BNE BNE BNE BNE BNE BNE BNE BN	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB33F ,X+ ,U+ LB32D	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A COMPARE TO STRING B CHECK ANOTHER CHARACTER IF =
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2741 2742 2741 2742	B310 B312 B314 B316 B319 B31D B31F B321 B325 B325 B327 B328 B32A B32C B32C B32D B32E B330 B332 B334 B336 B336 B337	D7 56 9F 58 9E 5F BD 86 96 56 34 04 AØ EØ 27 07 86 01 24 03 D6 56 40 DE 58 5C 5A 26 04 D6 56 44 D6 56 44 D6 56 44 D6 56 44 D6 56 45 50 50 50 50 50 50 50 50 50 50 50 50 50	59	* LB328 * ENTER W LB32D	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU LDB BNE LDB BNE LDB BNE LDB BNE LDB BRA LDA LDB BRA LDA LDB BEQ LDB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB334 FPØSGN LB337 ,X+ ,U+ LB320 #\$FF	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUIT OF RELATIONAL COMPARISON GET A BYTE FROM STRING B CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B
2721 2722 2723 2724 2725 2726 2727 2728 2729 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2744	B318 B312 B314 B316 B319 B321 B321 B321 B322 B322 B324 B326 B334 B336 B338 B338 B338 B338 B338 B338	D7 568 9F 589 9F	59	* LB328 * ENTER W LB32D	STB STX LDX JSR LDA JSR LDA BEQ LDA BECC LDB NEGA STITH ACCION LDB BRA LDB BRA LDB BRA BEQ LDB BRA BEQ LDB BCC	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB33F ,X+ ,U+ LB32D	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH  TRUE FLAG  TRUE IF LENGTH B > LENGTH A  LOAD ACCB WITH LENGTH B  SET FLAG = FALSE (1FF)  SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW  HORTER STRING  DECREMENT SHORTER STRING LENGTH  BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB  CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A  COMPARE TO STRING B  CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B  BRANCH IF STRING A > STRING B
2721 2722 2723 2724 2725 2726 2727 2728 2739 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2744 2745 2746 2747	B310 B312 B314 B316 B319 B31D B31F B321 B325 B325 B327 B328 B32A B32C B32C B32D B32E B330 B332 B334 B336 B336 B337	D7 568 9F 589 9F	59	* LB328 * ENTER W LB32D	STB STX LDX JSR LDA PSHS SUBA BEQ LDA BCC LDB NEGA STA LDU LDB BNE LDB BNE LDB BNE LDB BNE LDB BRA LDA LDB BRA LDA LDB BEQ LDB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB334 FPØSGN LB337 ,X+ ,U+ LB320 #\$FF	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B) = RETURN LENGTH AND ADDRESS OF STRING = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUIT OF RELATIONAL COMPARISON GET A BYTE FROM STRING B CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B
2721 2722 2723 2724 2725 2726 2727 2728 2729 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2744	B318 B312 B314 B316 B319 B321 B321 B321 B322 B322 B324 B326 B334 B336 B338 B338 B338 B338 B338 B338	D7 568 9F 589 9F	59	* LB328 * ENTER W LB32D LB334	STB STX LDX JSR LDA PSHS SUBA BEQ LDB NEGA STA LDU INCB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE BNE BNE BNE BNE BNE BNE BNE BNE BN	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB334 FPØSGN LB337 ,X+ ,U+ LB320 #\$FF	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A COMPARE TO STRING B CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B BRANCH IF STRING A > STRING B SET FLAG = TRUE
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2744 2745 2746 2747 2748	B310 B312 B314 B316 B319 B318 B323 B325 B327 B320 B320 B320 B320 B330 B330 B331 B331 B332 B332 B332 B333 B333 B334 B336 B338 B338 B338 B338 B338 B338 B338	D7 568 9F 589 9F	59	* LB328 * ENTER W LB32D LB334	STB STX LDX JSR LDA PSHS SUBA BEQ LDB NEGA STA LDU INCB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE BNE BNE BNE BNE BNE BNE BNE BNE BN	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB335 ,X+ ,U+ LB32D #\$FF LB32F LB33F	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A COMPARE TO STRING B CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B BRANCH IF STRING A > STRING B SET FLAG = TRUE
2721 2722 2723 2724 2725 2726 2727 2728 2739 2731 2732 2733 2734 2735 2736 2737 2738 2737 2749 2741 2742 2743 2744 2745 2746 2747 2748 2747 2748 2747 2748 2749 2759 2759	8318 8316 8319 8316 8318 8310 8321 8325 8327 8328 8320 8320 8320 8330 8334 8336 8336 8338 8338 8338	75 58 9E 5F 5B 86 9E 5F 5B 86 9E 5F 5B 86 9E 5F 5B 86 9E 5F 5B 5E	59	* LB328 * ENTER W LB32D LB334 * DETERMI	STB STX LDX JSR LDA JSR LDA BCQ LDB BCQ LDB BNE LDA LDI INCB BNE LDA LDU INCB BNE LDA LDB BCC LDB BNE LDA LDB BCC RE BRA LDB BCC BCR BCR BCR BCR BCR BCR BCR BCR BC	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB335 ,X+ ,U+ LB320 #\$FF LB33F	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A COMPARE TO STRING B CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B BRANCH IF STRING A > STRING B SET FLAG = TRUE  RESULT IN FPAØ CONVERT \$FF,Ø,1 TO Ø,1,2 NOW IT \$ 1,2,4 FOR > = <
2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2741 2742 2743 2744 2745 2746 2747 2748 2749 2751 2748 2749 2751	8318 8316 8319 8316 8318 8310 8321 8325 8327 8328 8320 8320 8320 8330 8334 8336 8336 8338 8338 8338	D7 568 9E 5F 8D 86 9E 5F 8D 8E 5E 5E 8D 8E 5E	59	* LB328 * ENTER W LB32D LB334 * DETERMI	STB STX LDX JSR LDA JSR LDA BCQ LDB BCQ LDB BNE LDA LDI INCB BNE LDA LDU INCB BNE LDA LDB BCC LDB BNE LDA LDB BCC RE BRA LDB BCC BCR BCR BCR BCR BCR BCR BCR BCR BC	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB335 ,X+ ,U+ LB32D #\$FF LB32F LB33F	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH  TRUE FLAG  TRUE IF LENGTH B > LENGTH A  LOAD ACCB WITH LENGTH B  SET FLAG = FALSE (1FF)  SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING  DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB  CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING B  CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B  BRANCH IF STRING A > B  BRANCH IF STRING A > STRING B  SET FLAG = TRUE  RESULT IN FPAØ  CONVERT \$FF, Ø, 1 TO Ø, 1, 2  NOW IT S 1, 2, 4 FOR > = <  AND THE ACTUAL COMPARISON WITH THE DESIRED -
2721 2722 2723 2724 2725 2726 2727 2728 2730 2731 2732 2733 2734 2735 2736 2737 2738 2740 2741 2742 2743 2744 2745 2746 2747 2748 2747 2748 2749 2750 2751	B310 B312 B314 B316 B319 B318 B310 B321 B327 B320 B327 B320 B320 B320 B320 B321 B321 B322 B334 B336 B338 B338 B338 B338 B338 B338 B338	D7 568 9E 5F 8D 86 9E 5F 8D 8E 5E 8D 8E	59	* LB328 * ENTER W LB32D LB334 * DETERMI	STB STX LDX JSR LDA PSHS SUBA BEQ LDB NEGA STA LDU INCB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDB LDB BNE LDB LDB BNE LDB LDB LDB LDB LDB LDB LDB LDB LDB LDB	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES  FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB337 ,X+ ,U+ LB320 #\$FF LB33F  TH OF COMPARISON - RETURN #1 RELFLG	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCA WITH LENGTH B  SET FLAG = FALSE (IFF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A  COMPARE TO STRING B  CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B  BRANCH IF STRING A > STRING B  SET FLAG = TRUE  RESULT IN FPAØ  CONVERT \$FF,Ø,1 TO Ø,1,2  NOW IT S 1,2,4 FOR > = < AND THE ACTUAL COMPARISON WITH THE DESIRED - COMPARISON
2721 2722 2723 2724 2725 2726 2727 2728 2739 2731 2732 2733 2734 2735 2736 2737 2738 2737 2748 2749 2741 2742 2743 2744 2745 2746 2747 2748 2749 2759 2759 2759 2759	B310 B312 B314 B316 B319 B318 B321 B327 B327 B328 B327 B328 B320 B327 B328 B320 B327 B338 B338 B338 B338 B338 B338 B338 B33	D7 56 89 5F 58 9E 5F 5B 86 9E 5F 5B 86 9E 5F 5B 5E	59	* LB328 * ENTER W LB32D LB334 * DETERMI	STB STX LDX JSR LDA LDA BEQ LDB BNE LDA LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDA BCQ LDB BNE LDA BEQ LDB BNE LDA BEQ LDB BNE LDA BEQ LDB BNE BNE LDA BEQ LDB BNE BNE BNE BNE BNE BNE BNE BNE BNE BN	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB33F ,X+ ,U+ LB32D #\$FF LB33F  TH OF COMPARISON - RETURN #1 RELFLG LB348	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A COMPARE TO STRING B CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B BRANCH IF STRING A > STRING B SET FLAG = TRUE  RESULT IN FPAØ CONVERT \$FF,Ø,1 TO Ø,1,2 NOW IT S 1,2,4 FOR > = < AND THE ACTUAL COMPARISON WITH THE DESIRED - COMPARISON BRANCH IF FALSE (NO MATCHING BITS)
2721 2722 2723 2724 2725 2726 2727 2728 2739 2731 2732 2733 2734 2735 2736 2737 2738 2739 2744 2745 2746 2747 2748 2749 2747 2748 2749 2751 2752 2753 2751 2752 2753 2751 2752 2753	B310 B312 B314 B316 B319 B318 B321 B321 B322 B322 B322 B328 B328 B328 B332 B332	D7 56 58 9E 5F 5B BB 66 9E 5F 68 9E 5E	59	* LB328  * ENTER W LB32D  LB334  * DETERMILLB33F	STB STX LDX JSR LDX JSR LDA BEQ LDB BNE LDB BRA LDA LDB BRC LDB BRA LDA LDB BRA LDA LDB BRA LDA LDB BRA LDA LDB BCQ LDB BCQ LDB BCD LDB LDB LDB LDB LDB LDB LDB LDB LDB LD	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB334 FPØSGN LB335 ,X+ ,U+ LB320 #\$FF LB325 #\$FF LB33F	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH  TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B  SET FLAG = FALSE (1FF)  SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW  HORTER STRING  DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING B  CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B  BRANCH IF STRING A > STRING B  SET FLAG = TRUE  RESULT IN FPAØ CONVERT \$FF, Ø, 1 TO Ø, 1, 2  NOW IT S 1, 2, 4 FOR > = <  AND THE ACTUAL COMPARISON WITH THE DESIRED - COMPARESON  BRANCH IF FALSE (NO MATCHING BITS)  TRUE FLAG
2721 2722 2723 2724 2725 2726 2727 2728 2739 2731 2732 2733 2734 2735 2736 2737 2738 2737 2748 2749 2741 2742 2743 2744 2745 2746 2747 2748 2749 2759 2759 2759 2759	B310 B312 B314 B316 B319 B318 B321 B321 B322 B322 B322 B328 B328 B328 B332 B332	D7 56 89 5F 58 9E 5F 5B 86 9E 5F 5B 86 9E 5F 5B 5E	59	* LB328 * ENTER W LB32D LB334 * DETERMI	STB STX LDX JSR LDA LDA BEQ LDB BNE LDA LDB BNE LDB BNE LDB BNE LDB BNE LDB BNE LDA BCQ LDB BNE LDA BEQ LDB BNE LDA BEQ LDB BNE LDA BEQ LDB BNE BNE LDA BEQ LDB BNE BNE BNE BNE BNE BNE BNE BNE BNE BN	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB33F ,X+ ,U+ LB32D #\$FF LB33F  TH OF COMPARISON - RETURN #1 RELFLG LB348	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B BRANCH IF STRINGS OF EQUAL LENGTH TRUE FLAG TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B SET FLAG = FALSE (1FF) SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW HORTER STRING DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING A COMPARE TO STRING B CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B BRANCH IF STRING A > STRING B SET FLAG = TRUE  RESULT IN FPAØ CONVERT \$FF,Ø,1 TO Ø,1,2 NOW IT S 1,2,4 FOR > = < AND THE ACTUAL COMPARISON WITH THE DESIRED - COMPARISON BRANCH IF FALSE (NO MATCHING BITS)
2721 2722 2723 2724 2725 2726 2727 2728 2730 2731 2732 2733 2734 2736 2737 2738 2740 2741 2742 2743 2744 2745 2746 2747 2748 2747 2748 2749 2750 2750 2750 2750 2750 2750 2750 2750	B310 B312 B314 B316 B319 B318 B321 B321 B322 B322 B322 B328 B328 B328 B332 B332	D7 56 58 9E 5F 5B BB 66 9E 5F 68 9E 5E	59	* LB328  * ENTER W LB32D  LB334  * DETERMILLB33F	STB STX LDX JSR LDX JSR LDA BEQ LDB BNE LDB BRA LDA LDB BRC LDB BRA LDA LDB BRA LDA LDB BRA LDA LDB BRA LDA LDB BCQ LDB BCQ LDB BCD LDB LDB LDB LDB LDB LDB LDB LDB LDB LD	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB334 FPØSGN LB335 ,X+ ,U+ LB320 #\$FF LB325 #\$FF LB33F	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH  TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B  SET FLAG = FALSE (1FF)  SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW  HORTER STRING  DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING B  CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B  BRANCH IF STRING A > STRING B  SET FLAG = TRUE  RESULT IN FPAØ CONVERT \$FF,0,1 TO 0,1,2 NOW IT S 1,2,4 FOR > = < AND THE ACTUAL COMPARISON WITH THE DESIRED - COMPARISON BRANCH IF FALSE (NO MATCHING BITS)  TRUE FLAG
2721 2722 2723 2724 2725 2726 2727 2728 2739 2731 2732 2733 2734 2735 2736 2737 2738 2744 2745 2740 2741 2742 2744 2745 2746 2747 2748 2749 2750 2750 2750 2750 2750 2750 2750 2750	B310 B312 B314 B316 B319 B31B B321 B323 B325 B327 B328 B320 B328 B320 B328 B334 B336 B336 B337 B338 B338 B338 B338 B338 B338 B338	D7 56 58 9E 5F 5B BB 66 9E 5F 68 9E 5E	59 7C	* LB328  * ENTER W LB32D  LB334  * DETERMINATION OF THE STATE OF THE S	STB STX LDX JSR LDX JSR LDA BEQ LDB BNE LDB BRA LDA LDB BRC LDB BRA LDA LDB BRA LDA LDB BRA LDA LDB BRA LDA LDB BCQ LDB BCQ LDB BCD LDB LDB LDB LDB LDB LDB LDB LDB LDB LD	STRDES STRDES+2 FPA1+2 LB659 STRDES B ,S+ LB328 #1 LB328 STRDES FPØSGN STRDES+2 CB CONTAINING LENGTH OF SH LB334 FPØSGN LB334 FPØSGN LB335 ,X+ ,U+ LB320 #\$FF LB325 #\$FF LB33F	GET LENGTH AND ADDRESS OF STRING WHOSE DESCRIPTOR ADDRESS IS IN THE BOTTOM OF FPAØ  * SAVE LENGTH AND ADDRESS IN TEMPORARY  * DESCRIPTOR (STRING B)  = RETURN LENGTH AND ADDRESS OF STRING  = WHOSE DESCRIPTOR ADDRESS IS STORED IN FPA1+2 LOAD ACCA WITH LENGTH OF STRING B  SAVE LENGTH A ON STACK SUBTRACT LENGTH A FROM LENGTH B  BRANCH IF STRINGS OF EQUAL LENGTH  TRUE FLAG  TRUE IF LENGTH B > LENGTH A LOAD ACCB WITH LENGTH B  SET FLAG = FALSE (1FF)  SAVE TRUE/FALSE FLAG POINT U TO START OF STRING COMPENSATE FOR THE DECB BELOW  HORTER STRING  DECREMENT SHORTER STRING LENGTH BRANCH IF ALL OF STRING NOT COMPARED GET TRUE/FALSE FLAB CHECK TRUTH OF RELATIONAL COMPARISON GET A BYTE FROM STRING B  CHECK ANOTHER CHARACTER IF = FALSE FLAG IF STRING A > B  BRANCH IF STRING A > STRING B  SET FLAG = TRUE  RESULT IN FPAØ CONVERT \$FF, Ø, 1 TO Ø, 1, 2  NOW IT S 1, 2, 4 FOR > = <  AND THE ACTUAL COMPARISON WITH THE DESIRED - COMPARESON  BRANCH IF FALSE (NO MATCHING BITS)  TRUE FLAG

2760	B34E	C6 Ø1		DIM	LDB	#1	DIMENSION FLAG
2761	B35Ø	8D Ø8			BSR	LB35A	SAVE ARRAY SPACE FOR THIS VARIABLE
2762	B352	9D A5			JSR	GETCCH	GET CURRENT INPUT CHARACTER
2763	B354	26 F5			BNE	LB34B	KEEP DIMENSIONING IF NOT END OF LINE
2764	B356	39			RTS		
2765						RIABLE - RETURN X AND	
2766 2767						NG TO VARIABLE DESCRIPTOR	DCT THO
2768						REQUIRES 7 BYTES - THE FILL VARIABLE NAME AND THE NEXT	
2769						DESCRIPTOR. IF BIT 7 OF THE	
2770						VARIABLE NAME IS SET, THE	
2771						DEF FN VARIABLE. IF BIT 7	0F
2772						TE OF VARIABLE NAME IS SET	
2773				* VARIAB	LE IS A	STRING, OTHERWISE THE VAR	IABLE
2774				* IS NUM			
2775						LE IS NOT FOUND, A ZERO VAI	RIABLE IS
2776						THE VARIABLE SPACE	
2777	B357	5 F		LB357	CLRB		DIMENSION FLAG = Ø; DO NOT SET UP AN ARRAY
2778	B358	9D A5		10254	JSR	GETCCH	GET CURRENT INPUT CHARACTER
2779 278Ø	B35A B35C	D7 Ø5 97 37		LB35A	STB STA	DIMFLG VARNAM	SAVE ARRAY FLAG SAVE INPUT CHARACTER
2781	B330	3/ 3/		* ENTDV		OR DEF FN VARIABLE SEARCH	SAVE INFUI CHARACIER
2782	B35E	9D A5		LB35C	JSR	GETCCH VARIABLE SEARCH	GET CURRENT INPUT CHARACTER
2783	B36Ø	8D 4Ø		25000	BSR	LB3A2	SET CARRY IF NOT ALPHA
2784	B362	10 25	FF 11		LBCS	LB277	SYNTAX ERROR IF NOT ALPHA
2785	B366	5 F			CLRB		DEFAULT 2ND VARIABLE CHARACTER TO ZERO
2786	B367	D7 Ø6			STB	VALTYP	SET VARIABLE TYPE TO NUMERIC
2787	B369	9D 9F			JSR	GETNCH	GET ANOTHER CHARACTER FROM BASIC
2788	B36B	25 Ø4			BCS	LB371	BRANCH IF NUMERIC (2ND CHARACTER IN
2789				*			VARIABLE MAY BE NUMERIC)
2790		8D 33			BSR	LB3A2	SET CARRY IF NOT ALPHA
2791	B36F	25 ØA			BCS	LB37B	BRANCH IF NOT ALPHA
2792	B371	1F 89		LB371	TFR	A,B	SAVE 2ND CHARACTER IN ACCB
2793 2794						ARACTERS UNTIL A NON ALPHA S FOUND – IGNORE ALL CHARA	
2795						AME AFTER THE 1ST TWO	CIERS
2796	B373	9D 9F		LB373	JSR	GETNCH	GET AN INPUT CHARACTER
2797		25 FC		20373	BCS	LB373	BRANCH IF NUMERIC
2798	B377	8D 29			BSR	LB3A2	SET CARRY IF NOT ALPHA
2799		24 F8			BCC	LB373	BRANCH IF ALPHA
2800	B37B	81 24		LB37B	CMPA	#'\$	CHECK FOR A STRING VARIABLE
2801	B37D	26 Ø6			BNE	LB385	BRANCH IF IT IS NOT A STRING
2802	B37F	Ø3 Ø6			COM	VALTYP	SET VARIABLE TYPE TO STRING
28Ø3	B381	CB 8Ø			ADDB	#\$80	SET BIT 7 OF 2ND CHARACTER (STRING)
28Ø4	B383	9D 9F			JSR	GETNCH	GET AN INPUT CHARACTER
28Ø5	B385	D7 38		LB385	STB	VARNAM+1	SAVE 2ND CHARACTER IN VARNAM+1
2806	B387	9A Ø8			ORA	ARYDIS	OR IN THE ARRAY DISABLE FLAG - IF = \$80,
2807	0000	00.00		*	CUDA	417	DON T SEARCH FOR VARIABLES IN THE ARRAYS
28Ø8 28Ø9 >	B389	8Ø 28 1Ø 27 (	aa 75		SUBA LBEQ	#'( LB404	IS THIS AN ARRAY VARIABLE? BRANCH IF IT IS
2810	B38F	ØF Ø8	כי שש		CLR	ARYDIS	RESET THE ARRAY DISABLE FLAG
2811	B391				LDX	VARTAB	POINT X TO THE START OF VARIABLES
2812	B393	DC 37			LDD	VARNAM	GET VARIABLE IN QUESTION
2813	B395	9C 1D		LB395	CMPX	ARYTAB	COMPARE X TO THE END OF VARIABLES
2814	B397	27 12			BEQ	LB3AB	BRANCH IF END OF VARIABLES
2815	B399	10 A3	81		CMPD	,X++	* COMPARE VARIABLE IN QUESTION TO CURRENT
2816		27 3E			BEQ	LB3DC	* VARIABLE AND BRANCH IF MATCH
2817	B39E	30 05			LEAX	5,X	= MOVE POINTER TO NEXT VARIABLE AND
2818	B3AØ	2Ø F3			BRA	LB395	= KEEP LOOKING
2819							
2820						NOT UPPER CASE ALPHA	
2821 2822		81 41		LB3A2	BCS	#'A LB3AA	* CARRY SET IF < A
2823		25 Ø4 8Ø 5B				#'Z+1	<u>-</u>
2824		8Ø A5				# Z+1 #-('Z+1)	= CARRY CLEAR IF <= 'Z'
2825		39		LB3AA	RTS	#-( 2·1)	- WARRI GEERR II - E
2826	507171	03				IABLE IN TABLE OF VARIABLES	S
2827	B3AB	8E ØØ 8	8A	LB3AB		#ZERO	POINT X TO ZERO LOCATION
2828	B3AE	EE E4			LDU	,S	GET CURRENT RETURN ADDRESS
2829	B3BØ	11 83 1	B2 87		CMPU	#LB287	DID WE COME FROM EVALUATE ALPHA EXPR ?
2830	B3B4	27 28			BEQ	LB3DE	YES - RETURN A ZERO VALUE
2831		DC 1F			LDD	ARYEND	* GET END OF ARRAYS ADDRESS AND
2832		DD 43			STD	V43	* SAVE IT AT V43
2833		C3 ØØ I	07			#7	= ADD 7 TO END OF ARRAYS (EACH
2834		DD 41			STD	V41	= VARIABLE = 7 BYTES) AND SAVE AT V41
2835		9E 1D			LDX	ARYTAB	* GET END OF VARIABLES AND SAVE AT V47
2836		9F 47	10		STX JSR	V47	* MAVE A SEVEN DVTE SIDT FOR NEW VARIABLE AT
2837 2838	рэсэ	BD AC	TC	*	USK	LAC1E	MAKE A SEVEN BYTE SLOT FOR NEW VARIABLE AT TOP OF VARIABLES
2839	B3C6	9E 41			LDX	V41	= GET NEW END OF ARRAYS AND SAVE IT
2840		9F 1F			STX	ARYEND	=
2841		9E 45			LDX	V45	* GET NEW END OF VARIABLES AND SAVE IT
2842		9F 1D			STX	ARYTAB	*
2843		9E 47			LDX	V47	GET OLD END OF VARIABLES
2844		DC 37			LDD	VARNAM	GET NEW VARIABLE NAME
2845		ED 81			STD	, χ++	SAVE VARIABLE NAME
2846	B3D4				CLRA		* ZERO OUT THE FP VALUE OF THE NUMERIC
2847	B3D5				CLRB	v	* VARIABLE OR THE LENGTH AND ADDRESS
2848	R3D6	ED 84			STD	, Х	* OF A STRING VARIABLE

2849	B3D8	ED Ø	2			STD	2,X	*
2850	B3DA	A7 Ø				STA	4,X	*
2851	B3DC	9F 3	9		LB3DC	STX	VARPTR	STORE ADDRESS OF VARIABLE VALUE
2852 2853	B3DE	39			LB3DE *	RTS		
2854	B3DF	90 8	ø øø	00 00	LB3DF	FCB	\$90,\$80,\$00,\$00,\$00	* FLOATING POINT -32768
2855					*		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	SMALLEST SIGNED TWO BYTE INTEGER
2856					*			
2857	B3E4	9D 9			LB3E4	JSR	GETNCH	GET AN INPUT CHARACTER FROM BASIC
2858	B3E6	BD B		1	LB3E6	JSR	LB141	GO EVALUATE NUMERIC EXPRESSION
2859	B3E9	96 5			LB3E9	LDA	FPØSGN	GET FPAØ MANTISSA SIGN
286Ø 2861	B3EB	2B 5	D			BMI	LB44A	FC ERROR IF NEGATIVE NUMBER
2862					* CONVERT	FPAØ '	TO A SIGNED TWO BYTE INTEG	ER: RETURN VALUE IN ACCD
2863	B3ED	BD B	1 43	3	INTCNV	JSR	LB143	TM ERROR IF STRING VARIABLE
2864	B3FØ	96 4	F			LDA	FPØEXP	GET FPAØ EXPONENT
2865	B3F2	81 9	Ø			CMPA	#\$90	* COMPARE TO 32768 - LARGEST INTEGER EXPONENT AND
2866	B3F4	25 Ø		_		BCS	LB3FE	* BRANCH IF FPAØ < 32768
2867	B3F6	8E B				LDX	#LB3DF	POINT X TO FP VALUE OF -32768
2868 2869	B3F9 B3FC	BD B 26 4		0		JSR BNE	LBC96 LB44A	COMPARE -32768 TO FPAØ FC ERROR IF NOT =
2870	B3FE	BD B		3	LB3FE	JSR	LBCC8	CONVERT FPAØ TO A TWO BYTE INTEGER
2871	B4Ø1	DC 5				LDD	FPAØ+2	GET THE INTEGER
2872	B4Ø3	39				RTS		
2873					* EVALUAT	E AN A	RRAY VARIABLE	
2874	B4Ø4	DC Ø			LB4Ø4	LDD	DIMFLG	GET ARRAY FLAG AND VARIABLE TYPE
2875	B4Ø6	34 Ø	6			PSHS	В,А	SAVE THEM ON STACK
2876 2877	B4Ø8 B4Ø9	12 5F				NOP CLRB		DEAD SPACE CAUSED BY 1.2 REVISION RESET DIMENSION COUNTER
2878	B4ØA	9E 3	7		LB4ØA	LDX	VARNAM	GET VARIABLE NAME
2879	B4ØC	34 1			25.57	PSHS	X,B	SAVE VARIABLE NAME AND DIMENSION COUNTER
288Ø	B4ØE	8D D	4			BSR	LB3E4	EVALUATE EXPRESSION (DIMENSION LENGTH)
2881	B410	35 3	4			PULS	B,X,Y	PULL OFF VARIABLE NAME, DIMENSION COUNTER,
2882					*			ARRAY FLAG
2883		9F 3				STX	VARNAM	SAVE VARIABLE NAME AND VARIABLE TYPE
2884 2885	B414 B416	DE 5				LDU PSHS	FPAØ+2 U,Y	GET DIMENSION LENGTH
2886	B418	5C	Ю			INCB	U, T	SAVE DIMENSION LENGTH, ARRAY FLAG, VARIABLE TYPE INCREASE DIMENSION COUNTER
2887	B419	9D A	5			JSR	GETCCH	GET CURRENT INPUT CHARACTER
2888	B41B	81 2				CMPA	#',	CHECK FOR ANOTHER DIMENSION
2889	B41D	27 E	В			BEQ	LB4ØA	BRANCH IF MORE
2890	B41F	D7 Ø				STB	TMPLOC	SAVE DIMENSION COUNTER
2891	B421	BD B		7		JSR	LB267	SYNTAX CHECK FOR A )
2892	B424	35 Ø				PULS	A,B	* RESTORE VARIABLE TYPE AND ARRAY
2893 2894	B426	DD Ø 9E 1				STD LDX	DIMFLG ARYTAB	* FLAG - LEAVE DIMENSION LENGTH ON STACK GET START OF ARRAYS
2895		90 1			LB42A	CMPX	ARYEND	COMPARE TO END OF ARRAYS
2896		27 2			25.27	BEQ	LB44F	BRANCH IF NO MATCH FOUND
2897	B42E	DC 3	7			LDD	VARNAM	GET VARIABLE IN QUESTION
2898	B43Ø	10 A	3 84	1		CMPD	, X	COMPARE TO CURRENT VARIABLE
2899	B433	27 Ø				BEQ	LB43B	BRANCH IF =
2900	B435	EC Ø				LDD	2,X	GET OFFSET TO NEXT ARRAY VARIABLE
29Ø1 29Ø2	B437 B439	3Ø 8 2Ø E				LEAX BRA	D,X LB42A	ADD TO CURRENT POINTER KEEP SEARCHING
2902	B43B	C6 1			LB43B	LDB	#2*9	REDIMENSIONED ARRAY ERROR
2904	B43D	96 Ø			25.05	LDA	DIMFLG	* TEST ARRAY FLAG - IF <>Ø YOU ARE TRYING
2905	B43F	26 Ø				BNE	LB44C	* TO REDIMENSION AN ARRAY
2906	B441	D6 Ø	3			LDB	TMPLOC	GET NUMBER OF DIMENSIONS IN ARRAY
2907	B443	E1 Ø				CMPB	4,X	COMPARE TO THIS ARRAYS DIMENSIONS
2908	B445	27 5				BEQ	LB4AØ	BRANCH IF =
2909	B447 B449	C6 1	И		LB447	LDB	#8*2	BAD SUBSCRIPT SKIP TWO BYTES
291Ø 2911		C6 Ø	R		LB44A	FCB LDB	SKP2 #4*2	ILLEGAL FUNCTION CALL
2912		7E A		5	LB44C	JMP	LAC46	JUMP TO ERROR SERVICING ROUTINE
2913								
2914					* INSERT	A NEW	ARRAY INTO ARRAY VARIABLES	
2915							RRAY VARIABLES IS PRECEEDE	
2916							K COMPOSED OF 5+2*N BYTES	
2917 2918							ENSIONS IN THE ARRAY. THE YTES Ø,1:VARIABLE S NAME;	
2919							S AND DESCRIPTOR BLOCK; 4:	
2920							ENGTH OF DIMENSION 1; 7,8:	
2921							5+N:LENGTH OF DIMENSION N.	
2922					,	·		
2923		CC Ø		5	LB44F	LDD	#5	* 5 BYTES/ARRAY ENTRY SAVE AT COEFPT
2924		DD 6				STD	COEFPT	*
2925		DC 3 ED 8				LDD	VARNAM	= GET NAME OF ARRAY AND SAVE IN
2926 2927		D6 Ø				STD LDB	,X TMPLOC	= FIRST 2 BYTES OF DESCRIPTOR GET NUMBER OF DIMENSIONS AND SAVE IN
2928		E7 Ø				STB	4,X	* 5TH BYTE OF DESCRIPTOR
2929		BD A		3		JSR	LAC33	CHECK FOR ROOM FOR DESCRIPTOR IN FREE RAM
2930		9F 4				STX	V41	TEMPORARILY SAVE DESCRIPTOR ADDRESS
2931		C6 Ø	В		LB461	LDB	#11	* DEFAULT DIMENSION VALUE:X(10)
2932	B463		_			CLRA		*
2933		ØD Ø				TST	DIMFLG	= CHECK ARRAY FLAG AND BRANCH IF
2934 2935		27 Ø 35 Ø				BEQ PULS	LB46D A R	= NOT DIMENSIONING AN ARRAY GET DIMENSION LENGTH
2935		C3 Ø		1		ADDD		ADD ONE (X(Ø) HAS A LENGTH OF ONE)
2937		ED Ø			LB46D	STD	5,X	SAVE LENGTH OF ARRAY DIMENSION

2938	B46F	8D	5 D				BSR	LB4CE	MULTIPLY ACCUM ARRAY SIZE NUMBER LENGTH
2939						*			OF NEW DIMENSION
2940	B471						STD	COEFPT	TEMP STORE NEW CURRENT ACCUMULATED ARRAY SIZE
2941	B473						LEAX	,	BUMP POINTER UP TWO
2942 2943	B475 B477						DEC BNE	TMPLOC LB461	* DECREMENT DIMENSION COUNTER AND BRANCH IF  * NOT DONE WITH ALL DIMENSIONS
	B479						STX	TEMPTR	SAVE ADDRESS OF (END OF ARRAY DESCRIPTOR - 5)
2945	B47B							TEMPTR	ADD TOTAL SIZE OF NEW ARRAY
2946	B47D			F7	C3		LBCS	LAC44	OM ERROR IF > \$FFFF
2947							TFR	D,X	SAVE END OF ARRAY IN X
2948 2949	B483 B486						JSR SUBD	LAC37 #STKBUF-5	MAKE SURE THERE IS ENOUGH FREE RAM FOR ARRAY SUBTRACT OUT THE (STACK BUFFER - 5)
2950	B489			33			STD	ARYEND	SAVE NEW END OF ARRAYS
2951	B48B						CLRA		ZERO = TERMINATOR BYTE
2952		30				LB48C	LEAX	-1,X	* STORE TWO TERMINATOR BYTES AT
2953	B48E						STA	5 , X	* THE END OF THE ARRAY DESCRIPTOR
	B49Ø B492						CMPX BNE	TEMPTR LB48C	*
2956	B494						LDX	V41	GET ADDRESS OF START OF DESCRIPTOR
2957							LDA	ARYEND	GET MSB OF END OF ARRAYS; LSB ALREADY THERE
2958	B498	93	41				SUBD	V41	SUBTRACT OUT ADDRESS OF START OF DESCRIPTOR
	B49A						STD	2,X	SAVE LENGTH OF (ARRAY AND DESCRIPTOR)
296Ø 2961	B49C B49E						LDA BNE	DIMFLG LB4CD	* GET ARRAY FLAG AND BRANCH * BACK IF DIMENSIONING
2961	B49E	20	20			* CALCIIIA		ITER TO CORRECT ELEMENT	^ BACK IF DIMENSIONING
2963	B4AØ	E6	Ø4			LB4AØ	LDB	4,X	GET THE NUMBER OF DIMENSIONS
2964	B4A2	D7	ØЗ				STB	TMPLOC	TEMPORARILY SAVE
	B4A4						CLRA		* INITIALIZE POINTER
2966	B4A5					1.0446	CLRB	COFFRE	* TO ZERO
2967 2968	B4A6 B4A8					LB4A6	STD PULS		SAVE ACCUMULATED POINTER  * PULL DIMENSION ARGUMENT OFF THE
	B4AA						STD	FPAØ+2	* STACK AND SAVE IT
297Ø	B4AC			Ø5			CMPD		COMPARE TO STORED DIM ARGUMENT
2971	B4AF						BCC	LB4EB	BS ERROR IF > = "DIM" ARGUMENT
2972							LDU	COEFPT	* GET ACCUMULATED POINTER AND
2973 2974	B4B3 B4B5						BEQ BSR	LB4B9 LB4CE	* BRANCH IF 1ST DIMENSION = MULTIPLY ACCUMULATED POINTER AND DIMENSION
2975	B4B7							FPAØ+2	= LENGTH AND ADD TO CURRENT ARGUMENT
2976	B4B9					LB4B9		2,X	MOVE POINTER TO NEXT DIMENSION
2977	B4BB	ØA	ØЗ				DEC	TMPLOC	* DECREMENT DIMENSION COUNTER AND
2978	B4BD	26	E7				BNE	LB4A6	* BRANCH IF ANY DIMENSIONS LEFT
2979 298Ø	B4BF	ED	E3			* MULITPL	STD	BY 5 - 5 BYTES/ARRAY VALUE ,S	
2981	B4C1		EJ				ASLB	,3	
2982							ROLA		TIMES 2
2983	B4C3	58					ASLB		
	B4C4						ROLA	_	TIMES 4
2985 2986	B4C5 B4C7	18 30						,S++ D,X	TIMES 5 ADD OFFSET TO START OF ARRAY
2987							LEAX	5,X	ADJUST POINTER FOR SIZE OF DESCRIPTOR
2988	B4CB						STX	VARPTR	SAVE POINTER TO ARRAY VALUE
2989	B4CD	39				LB4CD	RTS		
2990								E NUMBER IN 5,X BY THE 2 B	
2991 2992	B4CE	96	1α			LB4CE	. RETUR	N RESULT IN ACCD, BS ERROR #16	16 SHIFTS TO DO A MULTIPLY
	B4DØ					LD4CL	STA	V45	SHIFT COUNTER
	B4D2						LDD	5,X	* GET SIZE OF DIMENSION
2995	B4D4	DD	17				STD	BOTSTK	* AND SAVE IT
2996	B4D6	4 F					CLRA		* ZERO
2997 2998	B4D7 B4D8					LB4D8	CLRB		* ACCD = SHIFT ACCB LEFT
	B4D9					LD400	ROLA		= ONE BIT
	B4DA	25						LB4EB	'BS' ERROR IF CARRY
3001	B4DC							COEFPT+1	* SHIFT MULTIPLICAND LEFT ONE
	B4DE							COEFPT	* BIT - ADD MULTIPLIER TO ACCUMULATOR
	B4EØ B4E2								* IF CARRY <> Ø ADD MULTIPLIER TO ACCD
	B4E4						BCS		'BS' ERROR IF CARRY (>\$FFFF)
3006	B4E6					LB4E6	DEC	V45	* DECREMENT SHIFT COUNTER
	B4E8		ΕE					LB4D8	* IF NOT DONE
	B4EA						RTS		
3009 3010	B4EB	7 E	В4	47		LB4EB *	JMP	LB447	'BS' ERROR
3011						* MEM			
3012							NOT A	TRUE INDICATOR OF FREE MEM	ORY BECAUSE
3Ø13								A STKBUF SIZE BUFFER FOR	THE STACK
3014						* FOR WHI	CH MEM	DOES NOT ALLOW.	
3Ø15 3Ø16	B4EE	1 5	1 Ø			* MEM	TFR	S,D	PUT STACK POINTER INTO ACCD
3017						ALII		ARYEND	SUBTRACT END OF ARRAYS
	B4F2						FCB		SKIP ONE BYTE
3Ø19								UE IN ACCB INTO A FP NUMBE	
	B4F3	4 F					CLRA	LUE IN ACCO THTO : 5:0:=-::	CLEAR MS BYTE OF ACCD
3021	B4F4	αr	ae			* CONVERT GIVABF		LUE IN ACCD INTO A FLOATIN VALTYP	G POINT NUMBER IN FPAØ SET VARIABLE TYPE TO NUMERIC
	B4F4					GIVADE	STD	FPAØ	SAVE ACCD IN TOP OF FACA
							LDB	#\$90	EXPONENT REQUIRED IF THE TOP TWO BYTES
	B4F8	CO							
3Ø25 3Ø26	B4FA			_		*	JMP	LBC82	OF FPAØ ARE TO BE TREATED AS AN INTEGER IN FPAØ CONVERT THE REST OF FPAØ TO AN INTEGER

3027									
3Ø28 3Ø29	B4FD	D.D.	B1	42		* STR\$ STR	JSR	LB143	'TM' ERROR IF STRING VARIABLE
3030	B500		Ø3			SIK	LDU	#STRBUF+2	*CONVERT FP NUMBER TO ASCII STRING IN
3031	B5Ø3		BD				JSR	LBDDC	*THE STRING BUFFER
3Ø32	B5Ø6	32					LEAS	2,S	PURGE THE RETURN ADDRESS FROM THE STACK
3Ø33	B5Ø8	8E	Ø3	D8			LDX	#STRBUF+1	*POINT X TO STRING BUFFER AND SAVE
3Ø34	B5ØB	20	ØB				BRA	LB518	*THE STRING IN THE STRING SPACE
3Ø35								YTES OF STRING SPACE. RETU	IRN START
3036	DEAD	9 F	4.0					AND FRESPC	CAVE V TH WAR
3Ø37 3Ø38	B5ØD B5ØF	9F 8D				LB5ØD LB5ØF	STX BSR	V4D LB56D	SAVE X IN V4D RESERVE ACCB BYTES IN STRING SPACE
3039	B511	9 F				LB507	STX	STRDES+2	SAVE NEW STRING ADDRESS
3040	B513	D7				20011	STB	STRDES	SAVE LENGTH OF RESERVED BLOCK
3041	B515	39					RTS		
3Ø42	B516	30	1 F			LB516	LEAX	-1,X	MOVE POINTER BACK ONE
3Ø43								OM (X) UNTIL AN END OF LIN	
3Ø44									CHARAC OR ENDCHR IS MATCHED.
3Ø45 3Ø46								STRING IS STORED IN THE ST ART OF THE STRING IS <= ST	
3047	B518	86	22			LB518	LDA	#'"	* INITIALIZE
3048	B51A	97				20010	STA	CHARAC	* TERMINATORS
3Ø49	B51C	97				LB51A	STA	ENDCHR	* T0 "
3Ø5Ø	B51E	3Ø	Ø1			LB51E	LEAX	1,X	MOVE POINTER UP ONE
3Ø51	B52Ø	9 F					STX	RESSGN	TEMPORARILY SAVE START OF STRING
3Ø52	B522	9 F					STX	STRDES+2	SAVE START OF STRING IN TEMP DESCRIPTOR
3Ø53 3Ø54	B524 B526	C6 5C	FF			LB526	LDB INCB	#-1	INITIALIZE CHARACTER COUNTER TO - 1 INCREMENT CHARACTER COUNTER
3055	B527	A6	RØ			LD3Z0	LDA	, X+	GET CHARACTER
3056	B529	27					BEQ	LB537	BRANCH IF END OF LINE
3Ø57	B52B	91	Ø1				CMPA	CHARAC	* CHECK FOR TERMINATORS
3Ø58	B52D	27	Ø4				BEQ	LB533	* IN CHARAC AND ENDCHR
3Ø59	B52F	91					CMPA	ENDCHR	* DON T MOVE POINTER BACK
3060	B531	26					BNE	LB526	* ONE IF TERMINATOR IS "MATCHED"
3Ø61 3Ø62	B533 B535	81 27				LB533	CMPA BEQ	#'" LB539	= COMPARE CHARACTER TO STRING DELIMITER = & DON T MOVE POINTER BACK IF SO
3063	B537	30				LB537	LEAX	-1,X	MOVE POINTER BACK ONE
3064	B539	9 F				LB539	STX	COEFPT	SAVE END OF STRING ADDRESS
3Ø65	B53B	D7	56				STB	STRDES	SAVE STRING LENGTH IN TEMP DESCRIPTOR
3Ø66	B53D	DE	62				LDU	RESSGN	GET INITIAL STRING START
3Ø67	B53F			Ø3	D9		CMPU	#STRBUF+2	COMPARE TO START OF STRING BUFFER
3Ø68	B543	22				LB543	BHI	LB54C	BRANCH IF > START OF STRING BUFFER
3069	B545	8D					BSR	LB5ØD	GO RESERVE SPACE FOR THE STRING
3070 3071	B547 B549	9E BD	B6	45			LDX JSR	RESSGN LB645	POINT X TO THE BEGINNING OF THE STRING MOVE (B) BYTES FROM (X) TO
3072	5545	00	50	73		*	USIK	25043	[FRESPC] - MOVE STRING DATA
3Ø73						* PUT DIR	CT PAG	E STRING DESCRIPTOR BUFFER	
3074						* ON THE S	STRING	STACK. SET VARIABLE TYPE T	O STRING
3075	B54C	9 E				LB54C	LDX	TEMPPT	GET NEXT AVAILABLE STRING STACK DESCRIPTOR
3076	B54E		Ø1	D1			CMPX	#CFNBUF	COMPARE TO TOP OF STRING DESCRIPTOR STACK
3Ø77			Ø5				BNE	LB558	FORMULA O.K.
3079	B551		1 0				LDR		'STDING ENDMILLA TOO COMDLEY' EDDOD
3Ø78 3Ø79	B553	C6		46		LB555	LDB JMP	#15*2 LAC46	'STRING FORMULA TOO COMPLEX' ERROR JUMP TO ERROR SERVICING ROUTINE
3078 3079 3080		C6	AC	46		LB555 LB558	LDB JMP LDA	#15*2 LAC46 STRDES	'STRING FORMULA TOO COMPLEX' ERROR JUMP TO ERROR SERVICING ROUTINE * GET LENGTH OF STRING AND SAVE IT
3079	B553 B555	C6 7E	AC 56	46			JMP	LAC46	JUMP TO ERROR SERVICING ROUTINE
3Ø79 3Ø8Ø	B553 B555 B558 B55A B55C	C6 7E 96	AC 56 ØØ	46			JMP LDA STA LDD	LAC46 STRDES ,X STRDES+2	JUMP TO ERROR SERVICING ROUTINE * GET LENGTH OF STRING AND SAVE IT * IN BYTE Ø OF DESCRIPTOR = GET START ADDRESS OF ACTUAL STRING
3079 3080 3081 3082 3083	B553 B555 B558 B55A B55C B55E	C6 7E 96 A7 DC ED	AC 56 ØØ 58 Ø2	46			JMP LDA STA LDD STD	LAC46 STRDES ,X STRDES+2 2,X	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR
3079 3080 3081 3082 3083 3084	B553 B555 B558 B55A B55C B55E B56Ø	C6 7E 96 A7 DC ED 86	AC 56 ØØ 58 Ø2 FF	46			JMP LDA STA LDD STD LDA	LAC46 STRDES ,X STRDES+2 2,X #\$FF	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING
3079 3080 3081 3082 3083 3084 3085	B553 B555 B558 B55A B55C B55E B56Ø B562	C6 7E 96 A7 DC ED 86 97	AC 56 ØØ 58 Ø2 FF Ø6	46			JMP LDA STA LDD STD LDA STA	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG
3079 3080 3081 3082 3083 3084 3085 3086	B553 B555 B558 B55A B55C B55E B56Ø B562 B564	C6 7E 96 A7 DC ED 86 97	AC 56 ØØ 58 Ø2 FF Ø6 ØD	46			JMP LDA STA LDD STD LDA STA STX	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP LASTPT	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR
3079 3080 3081 3082 3083 3084 3085	B553 B555 B558 B55A B55C B55E B56Ø B562 B564 B566	C6 7E 96 A7 DC ED 86 97 9F	AC 56 ØØ 58 Ø2 FF Ø6 ØD 52	46			JMP LDA STA LDD STD LDA STA	LAC46 STRDES ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG
3079 3080 3081 3082 3083 3084 3085 3086 3087	B553 B555 B558 B55A B55C B55E B56Ø B562 B564	C6 7E 96 A7 DC ED 86 97 9F 9F 3Ø	AC 56 ØØ 58 Ø2 FF Ø6 ØD 52 Ø5	46			JMP LDA STA LDD STD LDA STA STX STX LEAX	LAC46 STRDES ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ
3079 3080 3081 3082 3083 3084 3085 3086 3087 3088 3089 3090	B553 B555 B558 B55A B55C B55E B560 B562 B564 B566 B566	C6 7E 96 A7 DC ED 86 97 9F 9F 3Ø 9F	AC 56 ØØ 58 Ø2 FF Ø6 ØD 52 Ø5	46		LB558	JMP LDA STA LDD STD LDA STA STX STX LEAX STX LEAX STX	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR
3079 3080 3081 3082 3083 3084 3085 3086 3087 3088 3089 3090 3091	B553 B555 B558 B55A B55C B55E B560 B562 B564 B566 B568 B568	C6 7E 96 A7 DC ED 86 97 9F 9F 3Ø 9F	AC 56 ØØ 58 Ø2 FF Ø6 ØD 52 Ø5	46		LB558  * RESERVE	JMP LDA STA LDD STD LDA STA STX STX LEAX STX RTS ACCB B	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAG+2 5,X TEMPPT YTES IN STRING STORAGE SPA	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR
3079 3080 3081 3082 3083 3084 3085 3085 3087 3088 3089 3090 3091 3092	B553 B555 B558 B55A B55C B55E B560 B562 B564 B566 B568 B568	C6 7E 96 A7 DC ED 86 97 9F 9F 3Ø 9F	AC 56 ØØ 58 Ø2 FF Ø6 ØD 52 Ø5	46		* RESERVE * RETURN I	JMP LDA STA LDD STD LDA STA STX STX STX LEAX STX RTS ACCB B	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR
3079 3080 3081 3082 3083 3084 3085 3086 3087 3088 3089 3090 3091 3092 3093	B553 B555 B558 B55A B55C B55E B56Ø B562 B564 B566 B568 B56A	C6 7E 96 A7 DC ED 86 97 9F 9F 3Ø 9F 39	AC 56 00 58 02 FF 06 0D 52 05 0B	46		* RESERVE * RETURN V * RESERVEI	JMP LDA STA LDD STD LDA STA STX STX LEAX STX RTS ACCB B NITH TH	LAC46 STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT YTES IN STRING STORAGE SPACE E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR
3079 3080 3081 3082 3083 3084 3085 3086 3087 3088 3099 3090 3091 3092 3093 3094	B553 B555 B558 B55A B55C B55C B56Ø B562 B564 B566 B568 B56A B56C	C6 7E 96 A7 DC ED 86 97 9F 30 9F 39	AC 56 00 58 02 FF 06 0D 52 05 0B	46		* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STX STX STX LEAX STX RTS ACCB B WITH TH CLR	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASIPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG
3079 3080 3081 3082 3083 3084 3085 3086 3087 3088 3089 3090 3091 3092 3093	B553 B555 B558 B55A B55C B55E B56Ø B562 B564 B566 B568 B56A	C6 7E 96 A7 DC ED 86 97 9F 3Ø 9F 39	AC 56 00 58 02 FF 06 0D 52 05 0B	46		* RESERVE * RETURN V * RESERVEI	JMP LDA STA LDD STD LDA STA STX STX LEAX STX RTS ACCB B NITH TH	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPACE STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR
3079 3080 3081 3082 3083 3084 3085 3086 3087 3088 3089 3090 3091 3092 3093 3093 3094 3095	B553 B555 B558 B55A B55C B55C B560 B562 B564 B566 B568 B56A B56A B56C	C6 7E 96 A7 DC ED 86 97 9F 30 9F 39 4F 34	AC 56 00 58 02 FFF 06 0D 52 05 0B 07 06	46		* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STX STX STX LEAX STX ACCB B NITH TH CLR CLRA	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPACE STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE
3079 3880 3081 3081 3083 3083 3085 3085 3086 3087 3089 3090 3091 3092 3093 3094 3095 3096 3097 3098	B553 B555 B558 B556 B556 B562 B566 B566 B566 B566 B566	C6 7E 96 A7 DC ED 86 97 9F 30 9F 39 4F 34 DC A3	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 E0			* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STA STX STX LEAX RTS ACCB B NITH TH O STRIN CLR CLRA PSHS LDD SUBD	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL B,A STRTAB ,S+	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK  GET START OF STRING VARIABLES  SUBTRACT STRING LENGTH
3079 3080 3081 3081 3082 3083 3084 3085 3087 3088 3099 3091 3092 3093 3094 3095 3096 3097 3098 3099	B553 B555 B558 B556 B55C B562 B564 B566 B566 B568 B56A B56C B56F B570 B577 B577 B5774 B576	C6 7E 96 A7 DC ED 86 97 9F 30 9F 39 4F 34 DC A3 10	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 E0 93			* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STA STX LEAX STX RTS ACCB B NITH TH STRIN CLR CLRA PSHS LDD CMPD	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL B,A STRTAB ,S+ FRETOP	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE
3079 30801 3081 3082 3083 3084 3085 3086 3087 3088 3090 3091 3092 3093 3094 3095 3096 3097 3098 3099 3100	B553 B555 B558 B556 B550 B550 B562 B564 B566 B566 B568 B56C B567 B572 B574 B574 B579	C6 7E 96 A7 DC ED 86 97 9F 30 9F 39 0F 4F 34 DC A3 10 25	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 0A			* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STA STX STX STX LEAX STX RTS ACCB B WITH TH OSTRIN CLR CLRA PSHS LDD CMPD BCS	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB ,S+ FRETOP LB585	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  MEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES  SUBTRACT STRING LENGTH COMPARE TO STARTO F STRING STORAGE  IF BELOW START, THEN REORGANIZE
3079 3080 3081 3081 3082 3083 3085 3085 3086 3091 3092 3093 3094 3095 3096 3097 3098 3099 3100 3101	B553 B555 B558 B556 B556 B562 B564 B566 B568 B566A B56C B56F B576 B577 B577 B577 B577	C6 7E 96 A7 DC ED 86 97 9F 30 9F 39 0F 4F A3 4 DC A3 10 25 DD	AC 56 00 58 2 FF 06 00 D 52 05 00 B 07 06 23 E0 93 0A 23			* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STA STX STX LEAX STX RTS ACCB B NITH TH O STRIN CLR CLRA LDD SUBD CMPD BCS STD	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB ,S+ FRETOP LB585	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE
3079 30801 3081 3082 3083 3084 3085 3086 3087 3088 3090 3091 3092 3093 3094 3095 3096 3097 3098 3099 3100	B553 B555 B558 B556 B5562 B5662 B5664 B5668 B566A B566C B566D B56FØ B572 B574 B579 B5779 B577B	06 7E 96 A7 DC ED 86 97 9F 39 9F 39 0F 4F 34 DC A3 10 25 DD 9E	AC 56 00 58 2 FF 06 00 D 52 05 00 B 07 06 33 00 A 23 23			* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STA STX STX LEAX STX RTS ACCB B NITH TH O STRIN CLR CLRA LDD SUBD CMPD BCS STD	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL B,A STRTAB STRTAB STRTAB STRTAB STRTAB	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK  GET START OF STRING VARIABLES  SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE  IF BELOW START, THEN REORGANIZE  SAVE NEW START OF STRING VARIABLES
3079 3080 3081 3081 3082 3083 3084 3085 3086 3087 3090 3090 3090 3091 3092 3093 3095 3096 3097 3098 3099 3101 3100 3101 3102	B553 B555 B558 B556 B556 B562 B566 B568 B566 B568 B566 B567 B570 B572 B576 B577 B577 B577 B577 B577	06 7E 96 A7 DC ED 86 97 9F 39 9F 34 DC A3 10 DD 9E 30 8	AC 56 00 58 02 FFF 06 D 52 05 0B 07 06 23 E0 93 0A 23 23 01			* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA STA LDD STD LDA STX STX STX STX LEAX STX RTS ACCB B ITH TH OSTRIN CLRA PSHS LDD CMPD BCS STD LDX	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB ,S+ FRETOP LB585 STRTAB 1,X	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  **CE  **CLEAR STRING REORGANIZATION FLAG  ** PUSH THE LENGTH OF THE  ** STRING ONTO THE STACK  GET START OF STRING VARIABLES  SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE  IF BELOW START, THEN REORGANIZE  SAVE NEW START, THEN REORGANIZE  SAVE NEW START, STRING VARIABLES  GET START OF STRING VARIABLES  GET START OF STRING VARIABLES
3079 3080 3081 3081 3082 3083 3084 3085 3086 3087 3089 3090 3090 3090 3091 3092 3093 3095 3096 3097 3098 3099 3100 3101 3102 3103 3104 3105	B553 B558 B558 B558 B556 B556 B566 B566 B566	C6 7E 96 A7 DC ED 97 9F 30 9F 39 4F 4F 34 10 25 DD 9E 30 97 35 39	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 04 23 23 01 25 84			* RESERVE * RESERVE * RESERVEI LB56D LB56F	JMP LDA LDA STD LDA STD LDA STX LEAX STX LEAX STX CLR CLR PSHS LDD BCS STD LDX LEAX LEAX LEAX LDD CMPD LDD LDX LEAX LEAX LDD LDX LDA	LAC46 STRDES , X STRDES+2 2, X #\$FF VALTYP LASTPT FPAØ+2 5, X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL B,A STRTAB ,S+ FRETOP LB585 STRTAB 1,X FRESPC B,PC	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  ACCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE  IF BELOW START, THEN REORGANIZE  GET START OF STRING VARIABLES  SAVE START ADDRESS OF NEWLY RESERVED SPACE  RESTORE NUMBER OF BYTES RESERVED AND RETURN
3079 30801 3081 3081 3082 3083 3084 3085 3086 3087 3090 3090 3091 3092 3093 3095 3096 3097 3100 3100 3100 3100 3100 3100 3100 310	B553 B555 B558 B558 B556 B556 B566 B566 B568 B568	C6 7E 96 A7 DC ED 97 9F 30 9F 39 4F 34 DC 25 DD 9E 30 9F 35 CC 66 CC 76 CC 76 76 76 76 76 76 76 76 76 76 76 76 76	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 04 23 23 14 1A			* RESERVE * RETURN N * RESERVEI LB56D	JMP LDA LDA LDD STD LDA STX	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPA0+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB 5,S+ FRETOP LB585 STRTAB 1,X FRESPC B,PC B,PC H2213	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH  COMPARE TO STARTO OF STRING STORAGE  IF BELOW START, THEN REORGANIZE  SAVE NEW START OF STRING VARIABLES  GET START OF STRING VARIABLES  GET START OF STRING VARIABLES  GET START OF STRING VARIABLES  ADD ONE  SAVE START ADDRESS OF NEWLY RESERVED SPACE  RESTORE NUMBER OF BYTES RESERVED AND RETURN  'OUT OF STRING SPACE' ERROR
3079 30801 3081 3082 3083 3084 3086 3087 3086 3089 3090 3091 3092 3093 3094 3095 3096 3097 3098 3100 3101 3102 3103 3104 3105	B553 B558 B558 B558 B556 B562 B568 B564 B568 B566 B568 B567 B577 B578 B579 B578 B578 B577 B578 B578 B578 B578 B578	06 7E 96 A7 DC ED 86 97 9F 39 9F 34 DC A3 10 9E 30 9F 35 66 03	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 01 25 84 1A 07			* RESERVE * RESERVE * RESERVEI LB56D LB56F	JMP LDA LDA STD LDA STD LDA STD LDA STX STX STX STX STX STX RTS STX RTS STX RTS LEAX RTS LEAX RTS STX RTS LEAX STX LEAX STX LDD LCA	LAC46 STRDES, ,X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB ,S+ FRETOP LB585 STRTAB 1,X FRESPC B,PC #2*13 GARBFL	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  MEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CCE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH COMPARE TO START OF STRING VARIABLES GET START STRING VARIABLES GET START STRING VARIABLES GET START STRING VARIABLES GET STRING SPACE' ERROR TOUT OF STRING SPACE' ERROR TOUT OF STRING SPACE' ERROR TOUT OF STRING SPACE' ERROR TOUGGLE REORGANIZATION FLAG
3079 3080 3081 3081 3082 3083 3084 3085 3086 3087 3089 3090 3090 3091 3092 3093 3095 3096 3097 3098 3099 3100 3101 3102 3103 3104 3105 3106 3107 3108	B553 B555 B558 B558 B556 B556 B566 B566 B566	C6 7E 96 A7 DC ED 86 97 9F 39 9F 39 DC A3 10 25 50 DD 9F 30 9F 50 50 50 50 50 50 50 50 50 50 50 50 50	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 03 23 23 125 84 1A 07 CA			* RESERVE * RESERVE * RESERVEI LB56D LB56F	JMP LDA LDA STD LDA STD LDA STSTA LEAX STX LEAX STX RTS ACCB B STLIN CLEAX STRIN CLEAX STRIN CLEAX LEAX LEAX LEAX LEAX LEAX LEAX LEAX	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL B,A STRTAB ,S+ FRETOP LBS85 STRTAB 1,X FRESPC B,PC #2*13 GARBFL LB555	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  **CCE  **CLEAR STRING REORGANIZATION FLAG  **PUSH THE LENGTH OF THE  ** STRING ONTO THE STACK  GET START OF STRING VARIABLES  SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE  IF BELOW START, THEN REORGANIZE  SAVE NEW START OF STRING VARIABLES  GET START ADDRESS OF NEWLY RESERVED SPACE  RESTORE NUMBER OF BYTES RESERVED AND RETURN  "OUT OF STRING SPACE" ERROR  TOGGLE REORGANIZATION FLAG  ERROR IF FRESHLY REORGANIZED
3079 3080 3081 3082 3083 30845 3085 3086 3087 3089 3090 3091 3092 3093 3094 3095 3096 3097 3100 3107 3106 3107 3108	B553 B558 B558 B558 B556 B556 B566 B568 B568	06 7E 96 A7 DC ED 86 97 9F 30 9F 34 DC 31 0 DC 9E 30 9F 35 C6 0 0 37 8D	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 08 23 12 58 4 1A 07 CA 04			* RESERVE * RESERVE * RESERVEI LB56D LB56F	JMP LDA LDA LDD STD LDA STD LDA STX	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL B,A STRTAB 5,S+ FRETOP LBS85 STRTAB 5,S+ FRETOP LBS86 STRTAB 1,X FRESPC B,PC #2*13 GARBFL LB555 LB551	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  **CE  **CLEAR STRING REORGANIZATION FLAG  ** PUSH THE LENGTH OF THE  ** STRING ONTO THE STACK  GET START OF STRING VARIABLES  SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE  IF BELOW START, THEN REORGANIZE  SAVE NEW START OF STRING VARIABLES  GET START OF STRING VARIABLES  GET START OF STRING VARIABLES  ADD ONE  SAVE NEW START ADDRESS OF NEWLY RESERVED SPACE  RESTORE NUMBER OF BYTES RESERVED AND RETURN  'OUT OF STRING SPACE' ERROR  TOGGLE REORGANIZATION FLAG  ERROR IF FRESHLY REORGANIZED  GO REORGANIZE STRING SPACE
3079 3080 3081 3081 3082 3083 3084 3085 3086 3087 3089 3090 3090 3091 3092 3093 3095 3096 3097 3098 3099 3100 3101 3102 3103 3104 3105 3106 3107 3108	B553 B555 B558 B558 B556 B556 B566 B566 B566	06 7E 96 A7 DC B6 97 9F 39 9F 39 9F 34 DC 31 0 25 DD 9E 35 C6 03 27 8D 35	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 09 3 A 23 23 1 25 4 1 A 7 C A 04 04			* RESERVE * RESERVE * RESERVEI LB56D LB56F	JMP LDA LDA LDD STD LDA STX	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPAØ+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL B,A STRTAB 5,S+ FRETOP LBS85 STRTAB 5,S+ FRETOP LBS86 STRTAB 1,X FRESPC B,PC #2*13 GARBFL LB555 LB551	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  **CCE  **CLEAR STRING REORGANIZATION FLAG  **PUSH THE LENGTH OF THE  ** STRING ONTO THE STACK  GET START OF STRING VARIABLES  SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE  IF BELOW START, THEN REORGANIZE  SAVE NEW START OF STRING VARIABLES  GET START ADDRESS OF NEWLY RESERVED SPACE  RESTORE NUMBER OF BYTES RESERVED AND RETURN  "OUT OF STRING SPACE" ERROR  TOGGLE REORGANIZATION FLAG  ERROR IF FRESHLY REORGANIZED
3079 30801 3081 3082 3083 3084 3085 3086 3087 3090 3091 3092 3093 3094 3095 3096 3097 3099 3100 3101 3102 3103 3104 3106 3107 3108 3109 31109	B553 B558 B558 B558 B556 B562 B568 B566 B568 B566 B568 B567 B570 B572 B574 B578 B578 B579 B578 B578 B578 B578 B578 B578 B578 B578	06 7E 96 A7 DC B6 97 9F 39 9F 39 9F 34 DC 31 0 25 DD 9E 35 C6 03 27 8D 35	AC 56 00 58 02 FF 06 0D 52 05 0B 07 06 23 09 3 A 23 23 1 25 4 1 A 7 C A 04 04			* RESERVE * RETURN I * RESERVEI LB56D LB56F	JMP LDA LDA LDD STD LDA STD LDA STX LEAX STX LEAX STX RTS SHITT STRIN CLRA PSHS SUBD CMPD LDX LEAX LEAX LEAX LEAX LEAX LEAX LEAX LEA	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPA0+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB STRTAB STRTAB 1,X FRESPC B,PC B,PC B,PC B,PC B,PC B,PC B,PC B,	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH COMPARE TO STARTO F STRING STORAGE IF BELOW START, THEN REORGANIZE SAVE NEW START OF STRING VARIABLES GET START OF STRING VARIABLES GET START OF STRING VARIABLES SUBTRACT STRING LENGTH COMPARE TO STARTO F STRING VARIABLES GET START OF STRING VARIABLES ADD ONE SAVE START ADDRESS OF NEWLY RESERVED SPACE RESTORE NUMBER OF BYTES RESERVED AND RETURN 'OUT OF STRING SPACE' ERROR TOGGLE REORGANIZATION FLAG ERROR IF FRESHLY REORGANIZED GO REORGANIZE STRING SPACE GET BACK THE NUMBER OF BYTES TO RESERVE
3079 30801 3081 3082 3083 30845 3086 3087 30886 3089 3090 3091 3092 3093 3094 3095 3096 3097 3100 3101 3106 3107 3108 3109 3110 31112 31113	B553 B558 B558 B558 B556 B566 B566 B566 B566	06 7E 96 A7 DC ED 86 97 39 9F 34 DC A3 30 25 DD 9E 35 C6 03 35 20 9E	AC 56 000 58 02 FF 6 00 D 52 05 B 07 06 23 08 23 1 25 84 A A 7 C A 4 4 D E 27			* RESERVE * RETURN N * RESERVEI LB56D LB56F  LB585  * REORGAN: LB591	JMP LDA LDA STD LDA STD LDA STX	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPA0+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB 5,S+ FRETOP LB585 STRTAB 1,X FRESPC B,PC #2*13 GARBFL LB555 B,PC #2*13 GARBFL LB555 B LB56F STRING SPACE MEMSIZ	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  CE  CLEAR STRING REORGANIZATION FLAG  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH COMPARE TO STARTO F STRING STORAGE IF BELOW START, THEN REORGANIZE SAVE NEW START OF STRING VARIABLES  GET START OF STRING VARIABLES  ADD ONE SAVE START ADDRESS OF NEWLY RESERVED SPACE RESTORE NUMBER OF BYTES RESERVED AND RETURN 'OUT OF STRING SPACE' ERROR TOGGLE REORGANIZATION FLAG  ERROR IF FRESHLY REORGANIZED GO REORGANIZE STRING SPACE  GET BACK THE NUMBER OF BYTES TO RESERVE TRY TO RESERVE ACCB BYTES AGAIN  GET THE TOP OF STRING SPACE
3079 3080 3081 3082 3083 3084 3085 3086 3087 3089 3090 3090 3091 3092 3093 3094 3095 3096 3097 3101 3102 3103 3104 3105 3106 3107 3108 3109 31101 31111 3112	B553 B555 B558 B558 B556 B556 B566 B566 B566	06 7E 96 A7 DC ED 86 A7 9F 39 9F 39 9F 34 DC A3 10 25 30 9F 35 36 27 9E 9F	AC 56 000 58 02 FF 6 00 D 52 05 B 07 06 23 08 23 1 25 84 A A 7 C A 4 4 D E 27			* RESERVE * RETURN I * RESERVEI LB56D LB56F	JMP LDA LDD STD LDA STD LDA STX STX STX STX STX STX STX RTS SS STX RTS SS SID CLEAX STX LDD CLEA LDD LDD LDA LDD LDD LDA LDD LDD LDD LD	LAC46 STRDES, X STRDES+2 2,X #\$FF VALTYP LASTPT FPA0+2 5,X TEMPPT  YTES IN STRING STORAGE SPA E STARTING ADDRESS OF THE G SPACE IN (X) AND FRESPC GARBFL  B,A STRTAB 5,S+ FRETOP LB585 STRTAB 1,X FRESPC B,PC #2*13 GARBFL LB555 B,PC #2*13 GARBFL LB555 B LB56F STRING SPACE MEMSIZ	JUMP TO ERROR SERVICING ROUTINE  * GET LENGTH OF STRING AND SAVE IT  * IN BYTE Ø OF DESCRIPTOR  = GET START ADDRESS OF ACTUAL STRING  = AND SAVE IN BYTES 2,3 OF DESCRIPTOR  * VARIABLE TYPE = STRING  * SAVE IN VARIABLE TYPE FLAG  = SAVE START OF DESCRIPTOR  = ADDRESS IN LASTPT AND FPAØ  5 BYTES/STRING DESCRIPTOR  NEXT AVAILABLE STRING VARIABLE DESCRIPTOR  * PUSH THE LENGTH OF THE  * STRING ONTO THE STACK GET START OF STRING VARIABLES SUBTRACT STRING LENGTH  COMPARE TO START OF STRING STORAGE  IF BELOW START, THEN REORGANIZE SAVE NEW START, THEN REORGANIZE SAVE NEW START OF STRING VARIABLES  GET START OF STRING VARIABLES  GET START OF STRING VARIABLES  GET START OF STRING VARIABLES  ADD ONE  SAVE SAVE START ADDRESS OF NEWLY RESERVED AND RETURN  'OUT OF STRING SPACE' ERROR  TOGGLE REORGANIZATION FLAG  ERROR IF FRESHLY REORGANIZED  GO REORGANIZE STRING SPACE  GET BACK THE NUMBER OF BYTES TO RESERVE  TRY TO RESERVE ACCB BYTES AGAIN

3116	B596	5F			CLRB		* AND RESET VARIABLE
3117	B597	DD 4B			STD	V4B	* POINTER TO Ø
3118	B599	9E 21			LDX	FRETOP	POINT X TO START OF STRING SPACE
3119	B59B	9F 47			STX	V47	SAVE POINTER IN V47
3120	B59D	8E Ø1	A9		LDX	#STRSTK	POINT X TO START OF STRING DESCRIPTOR STACK
3121	B5AØ	9C ØB		LB5AØ		TEMPPT	COMPARE TO ADDRESS OF NEXT AVAILABLE DESCRIPTOR
3122	B5A2	27 Ø4			BEQ	LB5A8	BRANCH IF TOP OF STRING STACK
3123	B5A4	8D 32			BSR	LB5D8	CHECK FOR STRING IN UNORGANIZED STRING SPACE
3124 3125	B5A6 B5A8	20 F8 9E 1B		LB5A8	BRA LDX	LB5AØ VARTAB	KEEP CHECKING GET THE END OF BASIC PROGRAM
3126	B5AA	9C 1D		LB5AA		ARYTAB	COMPARE TO END OF VARIABLES
3127	B5AC	27 Ø4		EDSAA	BEQ	LB5B2	BRANCH IF AT TOP OF VARIABLES
3128	B5AE	8D 22			BSR	LB5D2	CHECK FOR STRING IN UNORGANIZED STRING SPACE
3129	B5BØ	2Ø F8			BRA	LB5AA	KEEP CHECKING VARIABLES
3130	B5B2	9F 41		LB5B2	STX	V41	SAVE ADDRESS OF THE END OF VARIABLES
3131	B5B4	9E 41		LB5B4	LDX	V41	GET CURRENT ARRAY POINTER
3132	B5B6	9C 1F		LB5B6		ARYEND	COMPARE TO THE END OF ARRAYS
3133	B5B8	27 35			BEQ	LB5EF	BRANCH IF AT END OF ARRAYS
3134	B5BA	EC Ø2			LDD	2,X	GET LENGTH OF ARRAY AND DESCRIPTOR
3135	B5BC	D3 41			ADDD	V41	* ADD TO CURRENT ARRAY POINTER
3136 3137	B5BE B5CØ	DD 41			STD LDA	V41	* AND SAVE IT
3137	B5C2	A6 Ø1 2A FØ			BPL	1,X LB5B4	GET 1ST CHARACTER OF VARIABLE NAME BRANCH IF NUMERIC ARRAY
3139	B5C4	E6 Ø4			LDB	4,X	GET THE NUMBER OF DIMENSIONS IN THIS ARRAY
3140	B5C6	58			ASLB	.,.	MULTIPLY BY 2
3141	B5C7	CB Ø5			ADDB	#5	ADD FIVE BYTES (VARIABLE NAME, ARRAY
3142				*			LENGTH, NUMBER DIMENSIONS)
3143	B5C9	3A			ABX		X NOW POINTS TO START OF ARRAY ELEMENTS
3144	B5CA	9C 41		LB5CA	CMPX	V41	AT END OF THIS ARRAY?
3145	B5CC	27 E8			BEQ	LB5B6	YES - CHECK FOR ANOTHER
3146	B5CE	8D Ø8			BSR	LB5D8	CHECK FOR STRING LOCATED IN
3147				*			UNORGANIZED STRING SPACE
3148	B5DØ	2Ø F8			BRA	LB5CA	KEEP CHECKING ELEMENTS IN THIS ARRAY
3149	B5D2	A6 Ø1		LB5D2	LDA	1,X	GET F1RST BYTE OF VARIABLE NAME
3150	B5D4 B5D6	30 02 2A 14			LEAX BPL	2,X LB5EC	MOVE POINTER TO DESCRIPTOR
3151 3152	6300	ZM 14		+ CEADCH		RING - ENTER WITH X POINTIN	BRANCH IF VARIABLE IS NUMERIC
3152						CRIPTOR. IF STRING IS STOR	
3154						ID STRTAB, SAVE DESCRIPTOR	
3155						SET V47 TO STRING ADDRESS	70111211
3156	B5D8	E6 84		LB5D8	LDB	, Х	GET THE LENGTH OF THE STRING
3157	B5DA	27 10			BEQ	LB5EC	BRANCH IF NULL - NO STRING
3158	B5DC	EC Ø2			LDD	2,X	GET STARTING ADDRESS OF THE STRING
3159	B5DE	10 93	23		CMPD	STRTAB	COMPARE TO THE START OF STRING VARIABLES
3160	B5E1	22 Ø9			BHI	LB5EC	BRANCH IF THIS STRING IS STORED IN
3161				*			THE STRING VARIABLES
3162	B5E3	10 93	47		CMPD	V47	COMPARE TO START OF STRING SPACE
3163	B5E6	23 Ø4			BLS	LB5EC	BRANCH IF NOT STORED IN THE STRING SPACE
3164	B5E8	9F 4B			STX	V4B	SAVE VARIABLE POINTER IF STORED IN STRING SPACE
3165	B5EA	DD 47 30 05		LDEEC	STD	V47	SAVE STRING STARTING ADDRESS MOVE TO NEXT VARIABLE DESCRIPTOR
3166 3167	B5EC B5EE	39		LB5EC LB5EE	LEAX RTS	5,X	HOVE TO NEXT VARIABLE DESCRIPTOR
3168	B5EF	9E 4B		LB5EF	LDX	V4B	GET ADDRESS OF THE DESCRIPTOR FOR THE
3169	502.	J		*	25%		STRING WHICH IS STORED IN THE HIGHEST RAM ADDRESS IN
3170				*			THE UNORGANIZED STRING SPACE
3171	B5F1	27 FB			BEQ	LB5EE	BRANCH IF NONE FOUND AND REORGANIZATION DONE
3172	B5F3	4 F			CLRA		CLEAR MS BYTE OF LENGTH
3173	B5F4	E6 84			LDB	, Х	GET LENGTH OF STRING
3174	B5F6	5 A			DECB		SUBTRACT ONE
3175	B5F7	D3 47			ADDD	V47	ADD LENGTH OF STRING TO ITS STARTING ADDRESS
3176	B5F9	DD 43			STD	V43	SAVE AS MOVE STARTING ADDRESS
3177		9E 23			LDX	STRTAB	POINT X TO THE START OF ORGANIZED STRING VARIABLES
3178	B5FD	9F 41			STX	V41 LAC20	SAVE AS MOVE ENDING ADDRESS MOVE STRING FROM CURRENT POSITION TO THE
3179 318Ø	B5FF	BD AC	∠ W	*	JSR	LUC7A	TOP OF UNORGANIZED STRING SPACE
3181	B6Ø2	9E 4B			LDX	V4B	POINT X TO STRING DESCRIPTOR
3182		DC 45			LDD	V45	* GET NEW STARTING ADDRESS OF STRING AND
3183		ED Ø2			STD	2,X	* SAVE IT IN DESCRIPTOR
3184	B6Ø8	9E 45			LDX	V45	GET NEW TOP OF UNORGANIZED STRING SPACE
3185	B6ØA	3Ø 1F			LEAX		MOVE POINTER BACK ONE
3186 >	B6ØC	7E B5	93		JMP	LB593	JUMP BACK AND REORGANIZE SOME MORE
3187							
3188						IO STRINGS	
3189	B6ØF	DC 52		LB6ØF	LDD	FPAØ+2	* GET DESCRIPTOR ADDRESS OF STRING A
3190	B611	34 Ø6			PSHS	*	* AND SAVE IT ON THE STACK
3191	B613				JSR	LB223	GET DESCRIPTOR ADDRESS OF STRING B
3192 3193	B616 B619	BD B1 35 10			JSR PULS	LB146	'TM' ERROR IF NUMERIC VARIABLE * POINT X TO STRING A DESCRIPTOR
3193	B61B	9F 62			STX	RESSGN	* ADDRESS AND SAVE IT IN RESSGN
3194	B61D	E6 84			LDB	,X	GET LENGTH OF STRING A
3195		9E 52			LDX	,x FPAØ+2	POINT X TO DESCRIPTOR OF STRING B
3190		EB 84				, X	ADD LENGTH OF STRING B TO STRING A
3198		24 Ø5			BCC	, A LB62A	BRANCH IF LENGTH < 256
3199	B625	C6 1C			LDB	#2*14	'STRING TOO LONG' ERROR IF LENGTH > 255
3200	B627	7E AC			JMP	LAC46	JUMP TO ERROR SERVICING ROUTINE
3201	B62A	BD B5	ØD	LB62A	JSR	LB5ØD	RESERVE ROOM IN STRING SPACE FOR NEW STRING
3202	B62D	9E 62			LDX	RESSGN	GET DESCRIPTOR ADDRESS OF STRING A
3203		E6 84			LDB	, Х	GET LENGTH OF STRING A
		8D 10			BSR	LB643	MOVE STRING A INTO RESERVED BUFFER IN STRING SPACE
3204	B631	00 10					

3205	B633	Q.F.	4 D			LDX	V4D	GET DESCRIPTOR ADDRESS OF STRING B
3205	B635					BSR	LB659	GET LENGTH AND ADDRESS OF STRING B
3207	B637	8D				BSR	LB645	MOVE STRING B INTO REST OF RESERVED BUFFER
3208	B639					LDX	RESSGN	POINT X TO DESCRIPTOR OF STRING A
3209	B63B					BSR	LB659	DELETE STRING A IF LAST STRING ON STRING STACK
3210	B63D	BD	В5	4 C		JSR	LB54C	PUT STRING DESCRIPTOR ON THE STRING STACK
3211	B64Ø	7 E	В1	68		JMP	LB168	BRANCH BACK TO EXPRESSION EVALUATION
3212								
3213							FROM 2,X TO FRESPC	
3214	B643	AE			LB643	LDX	2,X	POINT X TO SOURCE ADDRESS
3215	B645		25		LB645	LDU	FRESPC	POINT U TO DESTINATION ADDRESS
3216 3217	B647 B648	5C	αı			INCB BRA	LB64E	COMPENSATION FOR THE DECB BELOW GO MOVE THE BYTES
3217	DU40	210	<b>8</b> 4		* MOVE B		ROM (X) TO (U)	do hove the bites
3219	B64A	Α6	80		LB64A	LDA	,X+	* GET A SOURCE BYTE AND MOVE IT
3220	B64C					STA	,U+	* TO THE DESTINATION
3221	B64E	5A			LB64E	DECB		DECREMENT BYTE COUNTER
3222	B64F	26	F9			BNE	LB64A	BRANCH IF ALL BYTES NOT MOVED
3223	B651		25			STU	FRESPC	SAVE ENDING ADDRESS IN FRESPC
3224	B653	39				RTS		
3225							(ACCB) AND ADDRESS (X) OF	
3226							DESCRIPTOR IS IN FPAØ+2	
3227 3228							RING IF IT IS THE LAST ONE RING STACK. REMOVE STRING F	TROM CTRING
3229							AT THE BOTTOM OF STRING V	
3230	B654	BD	B1	46	LB654	JSR	LB146	'TM' ERROR IF VARIABLE TYPE = NUMERIC
3231	B657				LB657	LDX	FPAØ+2	GET ADDRESS OF SELECTED STRING DESCRIPTOR
3232	B659	E6	84		LB659	LDB	, х	GET LENGTH OF STRING
3233	B65B	8D	18			BSR	LB675	* CHECK TO SEE IF THIS STRING DESCRIPTOR WAS
3234	B65D	26	13			BNE	LB672	* THE LAST ONE PUT ON THE STRING STACK AND
3235					*			* BRANCH IF NOT
3236	B65F					LDX	5+2,X	GET START ADDRESS OF STRING JUST REMOVED
3237		30					-1,X	MOVE POINTER DOWN ONE
3238 3239	B663 B665					CMPX BNE	STRTAB LB66F	COMPARE TO START OF STRING VARIABLES BRANCH IF THIS STRING IS NOT AT THE BOTTOM
3240	5000	20	סש		*	DIVE	LBOOF	OF STRING VARIABLES
3241	B667	34	Ø4			PSHS	В	SAVE LENGTH: ACCA WAS CLEARED
3242	B669					ADDD	STRTAB	* ADD THE LENGTH OF THE JUST REMOVED STRING
3243	B66B	DD				STD	STRTAB	* TO THE START OF STRING VARIABLES - THIS WILL
3244					*			* REMOVE THE STRING FROM THE STRING SPACE
3245	B66D	35	Ø4			PULS	В	RESTORE LENGTH
3246	B66F	3Ø	Ø1		LB66F	LEAX	1,X	ADD ONE TO POINTER
3247	B671	39				RTS		
3248	B672		02		LB672	LDX	2,X	*POINT X TO ADDRESS OF STRING NOT
3249	B674	39			* DEMOVE	RTS	FROM STRING STACK FATER I	*ON THE STRING STACK
325Ø 3251							FROM STRING STACK. ENTER W STRING DESCRIPTOR - DELETE	
3252							ACK IF IT IS ON TOP OF THE	
3253							STRING IS DELETED, SET THE	
3254	B675	9C	ØD		LB675		LASTPT	*COMPARE TO LAST USED DESCRIPTOR ADDRESS
3255	B677	26	Ø7			BNE	LB680	*ON THE STRING STACK, RETURN IF DESCRIPTOR
3256					*			*ADDRESS NOT ON THE STRING STACK
3257	B679	9 F				STX	TEMPPT	SAVE LAST USED DESCRIPTOR AS NEXT AVAILABLE
3258	B67B	30				LEAX	-5,X	* MOVE LAST USED DESCRIPTOR BACK 5 BYTES
3259	B67D	9F	טש			STX	LASTPT	* AND SAVE AS THE LAST USED DESCRIPTOR ADDR
326Ø 3261	B67F B68Ø	4F 39			LB68Ø	CLRA RTS		SET ZERO FLAG
3262	0000	33			LDOOD	KIS		
3263					* LEN			
3264	B681	8D	Ø3		LEN	BSR	LB686	POINT X TO PROPER STRING AND GET LENGTH
3265	B683	7 E	В4	F3	LB683	JMP	LB4F3	CONVERT ACCB TO FP NUMBER IN FPAØ
3266							RING ADDRESS LOAD LENGTH IN	
3267							TH THE STRING DESCRIPTOR I	I N
3268	0.000						ES OF FPAØ	OFT LENGTH AND ADDRESS OF STREET
	B686				LB686			GET LENGTH AND ADDRESS OF STRING SET VARIABLE TYPE TO NUMERIC
32/10						LLK	VALTYP	SEI VARIABLE IYPE IU NUMERIC
	B688					TCTD		
3271	B68A	5 D				TSTB		SET FLAGS ACCORDING TO LENGTH
3271 3272	B68A	5 D				TSTB RTS		
3271 3272 3273	B68A	5 D			* CHR\$			
3271 3272 3273 3274	B68A	5D 39		ØE	* CHR\$	RTS	LB7ØE	
3271 3272 3273 3274 3275 >	B68A B68B	5D 39 BD	В7			RTS		SET FLAGS ACCORDING TO LENGTH
3271 3272 3273 3274 3275 >	B68A B68B - B68C B68F B691	5D 39 BD C6 BD	B7 Ø1 B5	6D	CHR	JSR LDB		SET FLAGS ACCORDING TO LENGTH  CONVERT FPAØ TO AN INTEGER IN ACCD
3271 3272 3273 3274 3275 > 3276 3277 3278	B68A B68B - B68C B68F B691 B694	5D 39 BD C6 BD 96	B7 Ø1 B5 53	6D	CHR	JSR LDB JSR LDA	#1 LB56D FPAØ+3	SET FLAGS ACCORDING TO LENGTH  CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE
3271 3272 3273 3274 3275 > 3276 3277 3278 3279	B68A B68B B68C B68F B691 B694 B696	5D 39 BD C6 BD 96 BD	B7 Ø1 B5 53 B5	6D 11	CHR	JSR LDB JSR LDA JSR	#1 LB56D FPAØ+3 LB511	SET FLAGS ACCORDING TO LENGTH  CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR
3271 3272 3273 3274 3275 > 3276 3277 3278 3279 3280	B68A B68B B68C B68F B691 B694 B696 B699	5D 39 BD C6 BD 96 BD A7	B7 Ø1 B5 53 B5 84	6D 11	CHR LB68F	JSR LDB JSR LDA JSR STA	#1 LB56D FPAØ+3 LB511 ,X	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE)
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281	B68A B68B B68C B68F B691 B694 B696 B699 B69B	5D 39 BD C6 BD 96 BD A7 32	B7 Ø1 B5 53 B5 84 62	6D 11	CHR LB68F LB69B	JSR LDB JSR LDA JSR STA LEAS	#1 LB56D FPAØ+3 LB511 ,X 2,X	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282	B68A B68B B68C B68F B691 B694 B696 B699	5D 39 BD C6 BD 96 BD A7 32	B7 Ø1 B5 53 B5 84 62	6D 11	CHR LB68F	JSR LDB JSR LDA JSR STA	#1 LB56D FPAØ+3 LB511 ,X	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE)
3271 3272 3273 3274 3275 > 3276 3277 3278 3279 3280 3281 3282 3283	B68A B68B B68C B68F B691 B694 B696 B699 B69B	5D 39 BD C6 BD 96 BD A7 32	B7 Ø1 B5 53 B5 84 62	6D 11	CHR LB68F LB69B LB69D	JSR LDB JSR LDA JSR STA LEAS	#1 LB56D FPAØ+3 LB511 ,X 2,X	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284	B68A B68B B68C B68F B691 B694 B696 B699 B69B B69D	5D 39 BD C6 BD 96 BD A7 32 7E	B7 Ø1 B5 53 B5 84 62 B5	6D 11 4C	CHR LB68F LB69B LB69D * ASC\$	JSR LDB JSR LDA JSR STA LEAS	#1 LB56D FPA0+3 LB511 ,X 2,X LB54C	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK PUT TEMP DESCRIPTOR DATA ONTO STRING STACK
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3281 3282 3283 3284 3285	B68A B68B B68C B68F B691 B694 B696 B699 B69B B69D	5D 39 BD C6 BD 96 BD A7 32 7E	B7 Ø1 B5 53 B5 84 62 B5	6D 11 4C	CHR LB68F LB69B LB69D	JSR LDB JSR LDA JSR STA LEAS JMP	#1 LB56D FPAØ+3 LB511 ,X 2,X	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284 3284 3285 3286	B68A B68B - B68C B68F B691 B694 B696 B699 B69B B69D	5D 39 BD C6 BD 96 BD A7 32 7E	B7 Ø1 B5 53 B5 84 62 B5	6D 11 4C	CHR LB68F LB69B LB69D * ASC\$	JSR LDB JSR LDA JSR STA LEAS JMP	#1 LB56D FPAØ+3 LB511 ,X 2,X LB54C	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK PUT TEMP DESCRIPTOR DATA ONTO STRING STACK  PUT 1ST CHARACTER OF STRING INTO ACCB
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284 3284 3285 3286	B68A B68B - B68C B68F B691 B694 B696 B699 B69B B69D	5D 39  BD C6 BD 96 BD A7 322 7E  8D 20 8D 27	B7 Ø1 B5 53 B5 84 62 B5 Ø2 DF EØ 5E	6D 11 4C	CHR LB68F LB69B LB69D * ASC\$ ASC	JSR LDB JSR LDA JSR STA LEAS JMP	#1 LB56D FPAØ+3 LB511 ,X 2,X LB54C	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK PUT TEMP DESCRIPTOR DATA ONTO STRING STACK  PUT 1ST CHARACTER OF STRING INTO ACCB CONVERT ACCB INTO FP NUMBER IN FPAØ POINT X TO STRING DESCRIPTOR  *FC* ERROR IF NULL STRING
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284 3285 3286 3287 3286 3287	B68A B68B - B68C B68F B691 B694 B696 B699 B69B B69D B6AØ B6A2 B6A4 B6A6 B6A8	5D 39  BD C6 BD 96 BD A7 32 7E  8D 20 8D 27 E6	B7 Ø1 B5 53 B5 84 62 B5 Ø2 DF EØ 5E 84	6D 11 4C	CHR LB68F LB69B LB69D * ASC\$ ASC	JSR LDB JSR LDA JSR STA LEAS JMP BSR BSR BSR BSR BEQ LDB	#1 LB56D FPA0+3 LB511 ,X 2,X LB54C  LB6A4 LB683 LB686 LB706	SET FLAGS ACCORDING TO LENGTH  CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK PUT TEMP DESCRIPTOR DATA ONTO STRING STACK  PUT 1ST CHARACTER OF STRING INTO ACCB CONVERT ACCB INTO FP NUMBER IN FPAØ POINT X TO STRING DESCRIPTOR
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284 3285 3286 3287 3288 3289 3299	B68A B68B - B68C B68F B691 B694 B696 B699 B69B B69D	5D 39  BD C6 BD 96 BD A7 32 7E  8D 20 8D 27 E6	B7 Ø1 B5 53 B5 84 62 B5 Ø2 DF EØ 5E 84	6D 11 4C	CHR LB68F LB69B LB69D * ASC\$ ASC	JSR LDB JSR LDA JSR STA LEAS JMP BSR BSR BSR BEQ	#1 LB56D FPA0+3 LB511 ,X 2,X LB54C  LB6A4 LB683 LB686 LB706	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK PUT TEMP DESCRIPTOR DATA ONTO STRING STACK  PUT 1ST CHARACTER OF STRING INTO ACCB CONVERT ACCB INTO FP NUMBER IN FPAØ POINT X TO STRING DESCRIPTOR  *FC* ERROR IF NULL STRING
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284 3285 3286 3286 3287 3288 3289 3289 3289 3289	B68A B68B - B68C B68F B691 B694 B696 B699 B69B B69D B6AØ B6A2 B6A4 B6A6 B6A8	5D 39  BD C6 BD 96 BD A7 32 7E  8D 20 8D 27 E6	B7 Ø1 B5 53 B5 84 62 B5 Ø2 DF EØ 5E 84	6D 11 4C	CHR LB69B LB69D * ASC\$ ASC LB6A4	JSR LDB JSR LDA JSR STA LEAS JMP BSR BSR BSR BSR BEQ LDB	#1 LB56D FPA0+3 LB511 ,X 2,X LB54C  LB6A4 LB683 LB686 LB706	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK PUT TEMP DESCRIPTOR DATA ONTO STRING STACK  PUT 1ST CHARACTER OF STRING INTO ACCB CONVERT ACCB INTO FP NUMBER IN FPAØ POINT X TO STRING DESCRIPTOR  *FC* ERROR IF NULL STRING
3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284 3285 3286 3287 3288 3289 3299	B68A B68B - B68C B68F B691 B694 B696 B699 B69B B69D B6AØ B6A2 B6A4 B6A6 B6A8	5D 39  BD C6 BD 96 BD A7 32 7E  8D 20 8D 27 E6 39	B7 Ø1 B5 53 B5 84 62 B5 Ø2 DF EØ 58	6D 11 4C	CHR LB68F LB69B LB69D * ASC\$ ASC	JSR LDB JSR LDA JSR STA LEAS JMP BSR BSR BSR BSR BEQ LDB	#1 LB56D FPA0+3 LB511 ,X 2,X LB54C  LB6A4 LB683 LB686 LB706	CONVERT FPAØ TO AN INTEGER IN ACCD  * RESERVE ONE BYTE IN  * THE STRING SPACE GET ASCII STRING VALUE SAVE RESERVED STRING DESCRIPTOR IN TEMP DESCRIPTOR SAVE THE STRING (IT S ONLY ONE BYTE) PURGE THE RETURN ADDRESS OFF OF THE STACK PUT TEMP DESCRIPTOR DATA ONTO STRING STACK  PUT 1ST CHARACTER OF STRING INTO ACCB CONVERT ACCB INTO FP NUMBER IN FPAØ POINT X TO STRING DESCRIPTOR  *FC* ERROR IF NULL STRING

3294	B6AD	4 F			CLRA		CLEAR STRING POINTER OFFSET - OFFSET = Ø FOR LEFT\$
3295	B6AE	E1 84		LB6AE	CMPB	, х	* COMPARE LENGTH PARAMETER TO LENGTH OF
3296	B6BØ	23 Ø3			BLS	LB6B5	* STRING AND BRANCH IF LENGTH OF STRING
3297				*			>= LENGTH PARAMETER
3298	B6B2	E6 84			LDB	, X	USE LENGTH OF STRING OTHERWISE
3299	B6B4	4 F			CLRA		CLEAR STRING POINTER OFFSET (Ø FOR LEFT\$)
3300	B6B5	34 Ø6		LB6B5	PSHS	В,А	PUSH PARAMETERS ONTO STACK
3301	B6B7	BD B5	ØF		JSR	LB5ØF	RESERVE ACCB BYTES IN THE STRING SPACE
3302	B6BA	9E 4E	1		LDX	V4D	POINT X TO STRING DESCRIPTOR
33Ø3	B6BC	8D 9E			BSR	LB659	GET ADDRESS OF OLD STRING (X=ADDRESS)
33Ø4	B6BE	35 Ø4			PULS	В	* PULL STRING POINTER OFFSET OFF OF THE STACK
33Ø5	B6CØ	3A			ABX		* AND ADD IT TO STRING ADDRESS
33Ø6	B6C1	35 Ø4			PULS	В	PULL LENGTH PARAMETER OFF OF THE STACK
3307	B6C3	BD B6			JSR	LB645	MOVE ACCB BYTES FROM (X) TO [FRESPC]
3308	B6C6	2Ø D5	'		BRA	LB69D	PUT TEMP STRING DESCRIPTOR ONTO THE STRING STACK
3309				* RIGHT\$			
331Ø 3311	B6C8	8D 2E		RIGHT	BSR	LB6F5	GET ARGUMENTS FROM STACK
3312	B6CA	AØ 84		KIGIII	SUBA	,Х	ACCA=LENGTH PARAMETER - LENGTH OF OLD STRING
3313	B6CC	40			NEGA	, *	NOW ACCA = LENGTH OF OLD STRING
3314	B6CD	2Ø DF	:		BRA	LB6AE	PUT NEW STRING IN THE STRING SPACE
3315							
3316				* MID\$			
3317	B6CF	C6 FF		MID	LDB	#\$FF	* GET DEFAULT VALUE OF LENGTH AND
3318	B6D1	D7 53			STB	FPAØ+3	* SAVE IT IN FPAØ
3319	B6D3	9D AS	i		JSR	GETCCH	GET CURRENT CHARACTER FROM BASIC
3320	B6D5	81 29	1		CMPA	#')	ARGUMENT DELIMITER?
3321	B6D7	27 Ø5			BEQ	LB6DE	YES - NO LENGTH PARAMETER GIVEN
3322	B6D9	BD B2			JSR	LB26D	SYNTAX CHECK FOR COMMA
3323	B6DC	8D 2D			BSR	LB7ØB	EVALUATE NUMERIC EXPRESSION (LENGTH)
3324	B6DE	8D 15		LB6DE	BSR	LB6F5	GET ARGUMENTS FROM STACK
3325	B6EØ	27 24			BEQ	LB706	'FC' ERROR IF NULL STRING
3326	B6E2	5 F			CLRB		CLEAR LENGTH COUNTER (DEFAULT VALUE)
3327	B6E3	4A			DECA	V	*SUOTRACT ONE FROM POSITION PARAMETER (THESE
3328	B6E4	A1 84			CMPA	, X	*ROUTINES EXPECT 1ST POSITION TO BE ZERO, NOT ONE)
3329	DCEC	24 05		^	D.C.C	LDCDE	*AND COMPARE IT TO LENGTH OF OLD STRING
333Ø 3331	B6E6	24 CE	1		BCC	LB6B5	IF POSITION > LENGTH OF OLD STRING, THEN NEW STRING WILL BE A NULL STRING
3332	B6E8	1F 89			TFR	A,B	SAVE ABSOLUTE POSITION PARAMETER IN ACCB
3333	B6EA	EØ 84			SUBB	,χ	ACCB=POSITION-LENGTH OF OLD STRING
3334	B6EC	50			NEGB	,^	NOW ACCB=LENGTH OF OLDSTRING-POSITION
3335	B6ED	D1 53			CMPB	FPAØ+3	*IF THE AMOUNT OF OLD STRING TO THE RIGHT OF
3336	B6EF	23 C4			BLS	LB6B5	*POSITION IS <= THE LENGTH PARAMETER, BRANCH AND
3337				*USE ALL		STRING TO THE RIGHT OF THE	
3338						LENGTH PARAMETER	
3339	B6F1	D6 53			LDB	FPAØ+3	GET LENGTH OF NEW STRING
3340	B6F3	20 C	1		BRA	LB6B5	PUT NEW STRING IN STRING SPACE
3341				* DO A SY	NTAX CH	ECK FOR ")", THEN PULL THE	PREVIOUSLY CALCULATED NUMERIC
3342				* ARGUMEN	T (ACCD	) AND STRING ARGUMENT DESC	RIPTOR ADDR OFF OF THE STACK
3343	B6F5	BD B2	67	LB6F5	JSR	LB267	SYNTAX CHECK FOR A ")"
3344	B6F8	EE E4			LDU	, S	LOAD THE RETURN ADDRESS INTO U REGISTER
3345	B6FA	AE 65			LDX	5,8	* GET ADDRESS OF STRING AND
3346	B6FC	9F 4E			STX	V4D	* SAVE IT IN V4D
3347		A6 64			LDA	4,5	= PUT LENGTH OF STRING IN
	B6FE					4,\$	= BOTH ACCA AND ACCB
3348	B700	E6 64			LDB		
3349	B700 B702	32 67			LEAS	7,S	REMOVE DESCRIPTOR AND RETURN ADDRESS FROM STACK
3349 335Ø	B700 B702 B704	32 67 1F 35	i i	1 2 7 4 6	LEAS TFR	U,PC	JUMP TO ADDRESS IN U REGISTER
3349 3350 3351	B700 B702	32 67	i i	LB7Ø6 * FVALUAT	LEAS TFR JMP	U,PC LB44A	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL'
3349 3350 3351 3352	B700 B702 B704	32 67 1F 35	i i	* EVALUAT	LEAS TFR JMP E AN EX	U,PC LB44A PRESSION – RETURN AN INTEG	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL'
3349 3350 3351 3352 3353	B7ØØ B7Ø2 B7Ø4 B7Ø6	32 67 1F 35 7E B4	4A	* EVALUAT * ACCB -	LEAS TFR JMP E AN EX 'FC' ER	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN
3349 3350 3351 3352 3353 3354	B700 B702 B704 B706	32 67 1F 35 7E B4	4A	* EVALUAT * ACCB - LB7Ø9	LEAS TFR JMP E AN EX 'FC' ER JSR	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN GET NEXT BASIC INPUT CHARACTER
3349 3350 3351 3352 3353	B700 B702 B704 B706 B709 B70B	32 67 1F 35 7E B4	4A 41	* EVALUAT * ACCB -	LEAS TFR JMP E AN EX 'FC' ER	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN
3349 3350 3351 3352 3353 3354 3355	B700 B702 B704 B706 B709 B70B	32 67 1F 35 7E B4 9D 9F BD B1	4A 41	* EVALUAT * ACCB - LB709 LB70B	LEAS TFR JMP 'E AN EX 'FC' ER JSR JSR	U,PC LB44A PPRESSION - RETURN AN INTEG RORR IF EXPRESSION > 255 GETNCH LB141	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358	B700 B702 B704 B706 B709 B70B B70E B711 B712	32 67 1F 35 7E 84 9D 9F BD 81 BD 83 4D 26 F2	4A 41 E9	* EVALUAT * ACCB - LB709 LB70B	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255
3349 3350 3351 3352 3353 3354 3355 3356 3357	B700 B702 B704 B706 B709 B70B B70E B711 B712	32 67 1F 35 7E B4 9D 9F BD B1 BD B3 4D	4A 41 E9	* EVALUAT * ACCB - LB709 LB70B	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358	B700 B702 B704 B706 B709 B70B B70E B711 B712	32 67 1F 35 7E 84 9D 9F BD 81 BD 83 4D 26 F2	4A 41 E9	* EVALUAT * ACCB - LB709 LB70B LB70E	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255
3349 3350 3351 3352 3353 3354 3355 3355 3357 3358 3359 3360 3361	B700 B702 B704 B706 B709 B70B B70E B711 B712 B714	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5	4A 41 E9	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE JMP	U,PC LB44A PPRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3359 3360 3361 3362	B700 B702 B704 B706 B709 B708 B708 B711 B711 B7112 B714	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5	44 41 E9	* EVALUAT * ACCB - LB709 LB70B LB70E	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE JMP	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3359 3360 3361 3362 3363	B700 B702 B704 B706 B709 B708 B708 B711 B712 B714	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 8D 86 10 27	44A 41 E9	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE JMP	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ
3349 3350 3351 3352 3353 3354 3355 3356 3357 3368 3359 3360 3361 3362 3363 3363	B700 B702 B704 B706 B709 B708 B70E B711 B712 B714 B716 B719 B71D	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 BD 86 10 27 DE A6	44A 41 E9	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE JMP  JSR LBEQ LDU	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG PRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39 CHARAD	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U
3349 3350 3351 3352 3353 3354 3355 3356 3357 3360 3361 3361 3362 3363 3364 3365	B700 B702 B704 B706 B709 B708 B70E B711 B712 B714 B716 B719 B71D B71F	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 10 27 DE A6 9F A6	44A 41 E9	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3360 3361 3362 3363 3363 3364 3365 3365 3366	B700 B702 B704 B706 B709 B708 B70E B711 B712 B714 B716 B716 B719 B71D B71F B721	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 8D 86 9F A6 3A	44 41 E9 86 Ø3 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR JSR JSR JSR LBEQ LDU STX ABX	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG EROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39 CHARAD CHARAD	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR
3349 3350 3351 3352 3353 3354 3355 3357 3358 3359 3360 3361 3362 3363 3364 3365 3366 3366 3367	B700 B702 B704 B706 B709 B708 B711 B712 B714 B716 B719 B71D B71F B721 B722	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 8D 86 10 27 DE A6 3A A6 84	441 41 E9	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX ABX LDA	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9  LB7Ø6 GETCCH  LB686 LBA39 CHARAD CHARAD ,X	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT IMPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3359 3360 3361 3362 3363 3364 3365 3366 3366 3366 3366 3366	B700 B702 B704 B706 B709 B708 B711 B712 B714 B716 B719 B71D B71F B721 B722	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 8D 86 9F A6 3A	441 41 E9	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX ABX LDA	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG EROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39 CHARAD CHARAD	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCO TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3360 3361 3362 3363 3364 3365 3366 3367 3368 3369	B700 B702 B704 B706 B708 B708 B708 B711 B712 B714 B716 B716 B719 B71D B71F B721 B722 B724	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 8D 86 10 27 DE A6 9F A6 3A A6 84 34 52	4A 41 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JFR JFC EAN EX 'FC' EFF JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX ABX LDA PSHS	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39 CHARAD CHARAD ,X U,X,A	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING SET FING SAVE INPUT POINTER, STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3360 3361 3362 3363 3364 3365 3366 3367 3368 3369 3369 3369	B700 B702 B704 B706 B709 B708 B701 B711 B712 B714 B716 B719 B71D B71D B721 B722 B724 B726	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 8D 86 9F A6 3A A6 84 34 52 6F 84	441 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR TFR E AN EX FC' ER JSR JSR JSR TSTA BNE JMP  JSR LBEQ LDU LDU STX ABX LDA PSHS CLR	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG PRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCO TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER CLEAR STRING TERMINATOR : FOR ASCII - FP CONVERSION
3349 3350 3351 3352 3353 3354 3355 3366 3357 3361 3362 3363 3364 3365 3366 3367 3366 3367 3366 3367 3366 3370 3371	B700 B702 B704 B706 B709 B708 B70E B711 B712 B714 B716 B719 B71D B71F B722 B724 B726 B728	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 10 27 DE A6 34 52 6F 84 9D A5	44 41 E9	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX ABX LDA PSHS  CLR JSR	U,PC LB44A - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH  LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCO TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT IMPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR GET LAST BYTE OF STRING CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3360 3361 3362 3363 3364 3365 3366 3367 3368 3369 3369 3369	B700 B702 B704 B706 B709 B708 B701 B712 B714 B716 B716 B717 B721 B722 B724 B726 B728 B728 B724	32 67 1F 35 7E 84 9D 9F 8D 81 8D 83 4D 26 F2 ØE A5 8D 86 9F A6 3A A6 84 34 52 6F 84	44 41 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR TFR E AN EX FC' ER JSR JSR JSR TSTA BNE JMP  JSR LBEQ LDU LDU STX ABX LDA PSHS CLR	U,PC LB44A PRESSION - RETURN AN INTEG PROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH LBD12	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCO TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER CLEAR STRING TERMINATOR : FOR ASCII - FP CONVERSION
3349 3350 3351 3351 3352 3353 3354 3355 3356 3357 3360 3361 3362 3363 3365 3366 3367 3368 3369 3370 3370 3371 3372	8700 8702 8704 8706 8706 8708 8708 8711 8711 8714 8714 8715 8722 8724 8724 8726 8728 8724 8720	32 67 1F 35 7E 84 9D 9F 8D 83 4D 26 F2 ØE A5 9F A6 3A 34 52 6F 84 9D 86 6F 84 9D 86 8D 8D 86 8D	441 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JFR JMP E AN EX 'FC' ER JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX ABX LDA PSHS  CLR JSR JSR	U,PC LB44A PRESSION - RETURN AN INTEG PROR IF EXPRESSION > 255 GETNCH LB141 LB3E9 LB706 GETCCH LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH LBD12	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TO FLOATING POINT
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3360 3361 3362 3363 3364 3365 3366 3367 3368 3369 3370 3371 3372	8700 8702 8704 8706 8709 8708 8708 8711 8712 8714 8716 8719 8719 8719 8720 8724 8726 8728 8728 8720 8727 8727	32 67 1F 36 36 36 36 36 36 36 36 36 36 36 36 36	44 41 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX ABX LDA PSHS  CLR JSR PULS	U,PC LB44A PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9  LB706 GETCCH  LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH LBD12 A,X,U	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TERMINATOR POINT RESTORE CHARACTERS AND POINTERS
3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3360 3361 3362 3363 3364 3365 3366 3367 3368 3367 3368 3370 3371 3372 3373	8700 8702 8704 8706 8708 8708 8708 8711 8711 8714 8716 8716 8719 8710 8717 8722 8724	32 67 84 68 81 81 82 82 82 82 82 82 82 82 82 82 82 82 82	44 41 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX LDA PSHS  CLR JSR JSR CLR JSR JSR STA	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9  LB706 GETCCH  LB686 LBA39 CHARAD CHARAD CHARAD ,X U,X,A ,X GETCCH LB012 A,X,U ,X	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCO TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT IMPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR GET LAST BYTE OF STRING CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TO FLOATING POINT RESTORE CHARACTERS AND POINTERS REPLACE STRING TERMINATOR:
3349 3350 3351 3352 3353 3354 3355 3356 3357 3360 3363 3361 3362 3363 3363 3364 3365 3366 3367 3368 3370 3371 3372 3373	8700 8702 8704 8706 8706 8709 8708 8708 8711 8711 8712 8714 8716 8719 8716 8719 8721 8722 8724 8728 8728 8728 8728 8728 8728	32 67 84 68 81 81 82 82 82 82 82 82 82 82 82 82 82 82 82	44 41 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR JSR JSR JSR JSR LBEQ LDU STX ABX LDA PSHS CLR JSR JSR PSHS STU	U,PC LB44A - RETURN AN INTEG PRESSION - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9  LB706 GETCCH  LB686 LBA39 CHARAD CHARAD CHARAD ,X U,X,A ,X GETCCH LB012 A,X,U ,X	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCO TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT IMPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR GET LAST BYTE OF STRING CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TO FLOATING POINT RESTORE CHARACTERS AND POINTERS REPLACE STRING TERMINATOR:
3349 3350 3351 3352 3353 3355 3356 3357 3360 3361 3362 3363 3363 3363 3364 3365 3366 3367 3368 3369 3370 3371 3372 3373 3374 3375	8700 8702 8704 8706 8709 8708 8708 8708 8711 8712 8714 8716 8716 8717 8721 8722 8724 8728 8728 8728 8728 8733	32 677E 84  9D 9F 86 85 85 86 86 86 86 86 86 86 86 86 86 86 86 86	441 41 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL	LEAS TFR JMP E AN EX FC' ER JSR JSR JSR JSR JSR JSR LBEQ LDU STX ABX LDU STX A	U,PC LB44A - RETURN AN INTEG ROR IF EXPRESSION > 255 GETNCH LB141 LB3E9  LB706 GETCCH  LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH LB012 A,X,U ,X CHARAD LB73D	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT IMPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TO FLOATING POINT RESTORE CHARACTER AND POINTERS REPLACE STRING TERMINATOR RESTORE INPUT CHARACTER  * EVALUATE AN EXPRESSION, RETURN
3349 3350 3351 3351 3352 3353 3354 3355 3366 3357 3368 3361 3362 3363 3364 3365 3366 3367 3368 3370 3370 3370 3371 3372 3373 3375 3376 3377 3377	8700 8702 8704 8709 8706 8709 8708 8708 8712 8714 8716 8719 8716 8718 8721 8721 8722 8724 8728 8728 8728 8728 8731 8733	32 67 84 84 84 84 84 84 84 84 84 84 84 84 84	4A 41 E9 86 Ø3 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR JSR JSR JSR JSR STA BNE JMP  JSR LBEQ LDU STX ABX LDA PSHS CLR JSR PSHS CLR STX ABX LDA RTS STX	U,PC LB44A PRESSION - RETURN AN INTEG PROR IF EXPRESSION > 255 GETNCH LB141 LB3E9  LB706 GETCCH  LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH LB012 A,X,U ,X CHARAD LB73D BINVAL	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TO FLOATING POINT RESTORE CHARACTERS AND POINTERS REPLACE STRING TERMINATOR RESTORE INPUT CHARACTER  * EVALUATE AN EXPRESSION, RETURN * THE VALUE IN X; STORE IT IN BINVAL
3349 3350 3351 3352 3353 3354 3355 3356 3357 3360 3361 3362 3363 3364 3365 3366 3367 3368 3370 3371 3372 3373 3374 3375 3377 3378 3377 3378 3379 3380	8700 8702 8704 8706 8706 8708 8708 8711 8712 8714 8716 8719 8717 8718 8721 8722 8724 8728 8728 8728 8728 8728 8728	32 67 1F 33 7E 84 84 84 84 84 84 84 84 84 84 84 84 84	441 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL VAL	LEAS TFR JMP E AN EX 'FC' ER JSR JSR JSR JSR JSR TSTA BNE JMP  JSR LBEQ LDU STX ABX LDA PSHS CLR JSR PULS STA STU RTS BSR STX JSR	U,PC LB44A PRESSION - RETURN AN INTEG PRESSION > 255 GETNCH LB141 LB329 LB706 GETCCH LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH LBD12 A,X,U ,X CHARAD LB73D BINVAL LB26D	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT INPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR ADDRESS AND CHARACTER CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TO FLOATING POINT RESTORE CHARACTERS AND POINTERS REPLACE STRING TERMINATOR RESTORE INPUT CHARACTER  * EVALUATE AN EXPRESSION, RETURN * THE VALUE IN X; STORE IT IN BINVAL SYNTAX CHECK FOR A COMMA
3349 3350 3351 3351 3352 3353 3354 3355 3366 3357 3368 3361 3362 3363 3364 3365 3366 3367 3368 3370 3370 3370 3371 3372 3373 3375 3376 3377 3377 3377	8700 8702 8704 8706 8706 8708 8708 8711 8712 8714 8716 8719 8717 8718 8721 8722 8724 8728 8728 8728 8728 8728 8728	32 67 84 84 84 84 84 84 84 84 84 84 84 84 84	441 E9 86 03 1C	* EVALUAT  * ACCB - LB709 LB70B LB70E  * VAL VAL	LEAS TFR JMP E AN EX FC' ER JSR JSR JSR JSR JSR LBEQ LDU STX ABX LDU STX ABX LDA PSHS CLR JSR PULS STA STU RTS BSR STX	U,PC LB44A PRESSION - RETURN AN INTEG PROR IF EXPRESSION > 255 GETNCH LB141 LB3E9  LB706 GETCCH  LB686 LBA39 CHARAD CHARAD ,X U,X,A ,X GETCCH LB012 A,X,U ,X CHARAD LB73D BINVAL	JUMP TO ADDRESS IN U REGISTER 'ILLEGAL FUNCTION CALL' ER IN  GET NEXT BASIC INPUT CHARACTER EVALUATE A NUMERIC EXPRESSION CONVERT FPAØ TO INTEGER IN ACCD TEST MS BYTE OF INTEGER 'FC' ERROR IF EXPRESSION > 255 GET CURRENT INPUT CHARACTER FROM BASIC  POINT X TO STRING ADDRESS IF NULL STRING SET FPAØ SAVE INPUT POINTER IN REGISTER U POINT IMPUT POINTER TO ADDRESS OF STRING MOVE POINTER TO END OF STRING TERMINATOR GET LAST BYTE OF STRING SAVE INPUT POINTER, STRING TERMINATOR GET LAST BYTE OF STRING CLEAR STRING TERMINATOR: FOR ASCII - FP CONVERSION GET CURRENT CHARACTER FROM BASIC CONVERT AN ASCII STRING TO FLOATING POINT RESTORE CHARACTERS AND POINTERS REPLACE STRING TERMINATOR RESTORE CHARACTERS AND POINTERS REPLACE STRING TERMINATOR ** EVALUATE AN EXPRESSION, RETURN ** THE VALUE IN X; STORE IT IN BINVAL SYNTAX CHECK FOR A COMMA EVALUATE EXPRESSION IN RANGE Ø <= X < 256

3383				* EXPRESS	ION IS	NEGATIVE OR > 32767, I.E.	NOT A LEGAL POSITIVE INTEGER.
3384	B73D	BD B1	41	LB73D	JSR	LB141	EVALUATE NUMERIC EXPRESSION
3385	B74Ø	96 54		LB74Ø	LDA	FPØSGN	GET SIGN OF FPAØ MANTISSA
3386	B742	2B C2			BMI	LB7Ø6	'ILLEGAL FUNCTION CALL' IF NEGATIVE
3387		96 4F			LDA	FPØEXP	GET EXPONENT OF FPAØ
3388		81 90			CMPA		COMPARE TO LARGEST POSITIVE INTEGER
3389		22 BC	00		BHI	LB706	'ILLEGAL FUNCTION CALL' IF TOO LARGE
3390		BD BC	C8		JSR	LBCC8	SHIFT BINARY POINT TO EXTREME RIGHT OF FPAØ
3391 3392	B74D B74F	9E 52			LDX RTS	FPAØ+2	LOAD X WITH LOWER TWO BYTES OF FPAØ
3393	D/4F	39			KIS		
3394				* PEEK			
3395	B75Ø	8D EE		PEEK	BSR	LB740	CONVERT FPAØ TO INTEGER IN REGISTER X
3396		E6 84			LDB	, X	GET THE VALUE BEING 'PEEK'ED
3397		7E B4	F3		JMP	LB4F3	CONVERT ACCB INTO A FP NUMBER
3398							
3399				* POKE			
3400	B757	8D DB		POKE	BSR	LB734	EVALUATE 2 EXPRESSIONS
3401	B759	9E 2B			LDX	BINVAL	GET THE ADDRESS TO BE 'POKE'ED
3402		E7 84			STB	, X	STORE THE DATA IN THAT ADDRESS
3403	B75D	39			RTS		
3404				+ 11767			
34Ø5 34Ø6	D7EE	C6 FE		* LLIST LLIST	LDB	#-2	* SET DEVICE NUMBER TO
3407		D7 6F		LLISI	STB	DEVNUM	* PRINTER
3408		9D A5			JSR	GETCCH	GET CURRENT CHARACTER FROM BASIC
3409							
3410				* LIST			
3411	B764	34 Ø1		LIST	PSHS	CC	SAVE ZERO FLAG ON STACK
3412	B766	BD AF	67		JSR	LAF67	CONVERT DECIMAL LINE NUMBER TO BINARY
3413	B769	BD AD	Ø1		JSR	LADØ1	* FIND RAM ADDRESS OF THAT LINE NUMBER AND
3414	B76C	9F 66			STX	LSTTXT	* SAVE IT IN LSTTXT
3415		35 Ø1			PULS	CC	GET ZERO FLAG FROM STACK
3416		27 12			BEQ	LB784	BRANCH IF END OF LINE
3417		9D A5			JSR	GETCCH	GET CURRENT CHARACTER FROM BASIC
3418		27 13			BEQ	LB789	BRANCH IF END OF LINE
3419		81 AC			CMPA		MINUS TOKEN (IS IT A RANGE OF LINE NUMBERS?)
342Ø 3421		26 Ø9 9D 9F			BNE JSR	LB783 GETNCH	NO - RETURN GET NEXT CHARACTER FROM BASIC
3422		27 Ø6			BEQ	LB784	BRANCH IF END OF LINE
3423		BD AF	67		JSR	LAF67	GET ENDING LINE NUMBER
3424		27 Ø6			BEQ	LB789	BRANCH IF LEGAL LINE NUMBER
3425	B783	39		LB783	RTS		
3426				* LIST TH	E ENTIF	RE PROGRAM	
3427		CE FF	FF	LB784	LDU	#\$FFFF	* SET THE DEFAULT ENDING LINE NUMBER
3428	B787	DF 2B			STU	BINVAL	* TO \$FFFF
3429		32 62		LB789	LEAS		PURGE RETURN ADDRESS FROM THE STACK
3430		9E 66			LDX	LSTTXT	POINT X TO STARTING LINE ADDRESS
3431		BD B9		LB78D	JSR	LB95C	MOVE CURSOR TO START OF A NEW LINE
3432 3433		BD A5 EC 84	49		JSR LDD	LA549 ,X	CHECK FOR A BREAK OR PAUSE GET ADDRESS OF NEXT BASIC LINE
3434		26 Ø8			BNE	LB79F	BRANCH IF NOT END OF PROGRAM
3435		BD A4	2D	LB797	JSR	LA42D	CHECK CLOSE FILE HANDLER
3436		ØF 6F		20737	CLR	DEVNUM	SET DEVICE NUMBER TO SCREEN
3437		7E AC	73		JMP	LAC73	RETURN TO BASIC S MAIN INPUT LOOP
3438	B79F	9F 66		LB79F	STX	LSTTXT	SAVE NEW STARTING LINE ADDRESS
3439	B7A1	EC Ø2			LDD	2,X	* GET THE LINE NUMBER OF THIS LINE AND
3440	B7A3	10 93	2B		CMPD	BINVAL	* COMPARE IT TO ENDING LINE NUMBER
3441		22 EF			BHI	LB797	EXIT IF LINE NUMBER > ENDING LINE NUMBER
3442		BD BD			JSR	LBDCC	PRINT THE NUMBER IN ACCD ON SCREEN IN DECIMAL
3443	B7AB	BD B9	AC		JSR	LB9AC	SEND A SPACE TO CONSOLE OUT
3444 3445		9E 66 8D 1Ø			LDX BSR	LSTTXT LB7C2	GET RAM ADDRESS OF THIS LINE UNCRUNCH A LINE
3446		AE 9F	99 66		LDX	[LSTTXT]	POINT X TO START OF NEXT LINE
3447		CE Ø2			LDU	#LINBUF+1	POINT U TO BUFFER FULL OF UNCRUNCHED LINE
3448		A6 CØ	-	LB7B9	LDA	,U+	GET A BYTE FROM THE BUFFER
3449	B7BB	27 DØ			BEQ	LB78D	BRANCH IF END OF BUFFER
3450	B7BD	BD B9	B1		JSR	LB9B1	SEND CHARACTER TO CONSOLE OUT
3451	B7CØ	2Ø F7			BRA	LB7B9	GET ANOTHER CHARACTER
3452							
3453						NE INTO BASIC S LINE INPUT	
3454		BD Ø1	A6	LB7C2	JSR	RVEC24	HOOK INTO RAM
3455		30 04	a2 P2		LEAX	•	MOVE POINTER PAST ADDRESS OF NEXT LINE AND LINE NUMBER
3456		10 8E	שט שט	10700	LDY	#LINBUF+1	UNCRUNCH LINE INTO LINE INPUT BUFFER
3457		A6 8Ø 27 51		LB7CB	LDA BEQ	, X+	GET A CHARACTER BRANCH IF END OF LINE
3458 3459		27 51 2B 15			BMI	LB820 LB7E6	BRANCH IF END OF LINE BRANCH IF IT S A TOKEN
3459 346Ø		2B 15 81 3A			CMPA		CHECK FOR END OF SUB LINE
3461		26 ØD			BNE	# : LB7E2	BRNCH IF NOT END OF SUB LINE
3462		E6 84			LDB	,X	GET CHARACTER FOLLOWING COLON
3463		C1 84				, x #\$84	TOKEN FOR ELSE?
3464		27 FØ			BEQ	LB7CB	YES - DON T PUT IT IN BUFFER
3465		C1 83				#\$83	TOKEN FOR REMARK?
3466		27 EC			BEQ	LB7CB	YES - DON T PUT IT IN BUFFER
3467	B7DF				FCB	SKP2	SKIP TWO BYTES
3468		86 21		LB7EØ	LDA	#'!	EXCLAMATION POINT
3469		8D 3Ø		LB7E2	BSR	LB814	PUT CHARACTER IN BUFFER
347Ø 3471	B/E4	2Ø E5		* UNCRUNC	BRA A TOP	LB7CB	GET ANOTHER CHARACTER
J4/1				ONCRUNC	101	XLII	

3472	B7E6	CE Ø1 :	16	LB7E6	LDU	#COMVEC-10	FIRST DO COMMANDS
3473	B7E9	81 FF	10	LD7 LO	CMPA	#\$FF	CHECK FOR SECONDARY TOKEN
3474	B7EB	26 Ø4			BNE	LB7F1	BRANCH IF NON SECONDARY TOKEN
3475	B7ED	A6 8Ø			LDA	, X+	GET SECONDARY TOKEN
3476 3477	B7EF B7F1	33 45 84 7F		LB7F1	LEAU ANDA	5,U #\$7F	BUMP IT UP TO SECONDARY FUNCTIONS MASK OFF BIT 7 OF TOKEN
3478	B7F3	33 4A		LB7F3	LEAU	10,U	MOVE TO NEXT COMMAND TABLE
3479	B7F5	6D C4			TST	, U	IS THIS TABLE ENABLED?
3480	B7F7	27 E7			BEQ	LB7EØ	NO - ILLEGAL TOKEN
3481 3482	B7F9 B7FB	AØ C4 2A F6			SUBA BPL	,U LB7F3	SUBTRACT THE NUMBER OF TOKENS FROM THE CURRENT TOKEN NUMBER BRANCH IF TOKEN NOT IN THIS TABLE
3483	B7FD	AB C4			ADDA	,U	RESTORE TOKEN NUMBER RELATIVE TO THIS TABLE
3484	B7FF	EE 41			LDU	1,U	POINT U TO COMMAND DICTIONARY TABLE
3485	B8Ø1	4A		LB8Ø1	DECA	1.0004	DECREMENT TOKEN NUMBER
3486 3487	B8Ø2	2B Ø6		* CKID TH	BMI ROUGH D	LB80A ICTIONARY TABLE TO START (	BRANCH IF THIS IS THE CORRECT TOKEN
3488	B8Ø4	6D CØ		LB8Ø4	TST	,U+	GRAB A BYTE
3489	B8Ø6	2A FC			BPL	LB8Ø4	BRANCH IF BIT 7 NOT SET
3490	B8Ø8	2Ø F7			BRA	LB8Ø1	GO SEE IF THIS IS THE CORRECT TOKEN
3491 3492	B8ØA B8ØC	A6 C4 8D Ø6		LB8ØA	LDA BSR	,U LB814	GET A CHARACTER FROM DICTIONARY TABLE PUT CHARACTER IN BUFFER
3493	B8ØE	6D CØ			TST	,U+	CHECK FOR START OF NEXT TOKEN
3494	B81Ø	2A F8			BPL	LB8ØA	BRANCH IF NOT DONE WITH THIS TOKEN
3495	B812	20 B7			BRA	LB7CB	GO GET ANOTHER CHARACTER
3496 3497	B814	10 8C 6	Ø3 D6	LB814	CMPY BCC	#LINBUF+LBUFMX	TEST FOR END OF LINE INPUT BUFFER BRANCH IF AT END OF BUFFER
3498	B818 B81A	24 Ø6 84 7F			ANDA	LB820 #\$7F	MASK OFF BIT 7
3499	B81C	A7 AØ			STA	, Y+	* SAVE CHARACTER IN BUFFER AND
3500	B81E	6F A4			CLR	, Υ	* CLEAR NEXT CHARACTER SLOT IN BUFFER
3501	B82Ø	39		LB82Ø *	RTS		
35Ø2 35Ø3					THE LIN	E THAT THE INPUT POINTER	I\$
3504						TO THE LINE INPUT BUFFER	
35Ø5				* RETURN	LENGTH	OF CRUNCHED LINE IN ACCD	
3506	0001	DD 01		*	160	DUECOO	UOOV THTO DAM
35Ø7 35Ø8	B821	BD Ø1 / 9E A6	A3	LB821	JSR LDX	RVEC23 CHARAD	HOOK INTO RAM GET BASIC'S INPUT POINTER ADDRESS
3509	B826	CE Ø2 I	DC		LDV	#LINBUF	POINT X TO LINE INPUT BUFFER
3510	B829	ØF 43		LB829	CLR	V43	CLEAR ILLEGAL TOKEN FLAG
3511	B82B	ØF 44			CLR	V44	CLEAR DATA FLAG
3512 3513	B82D B82F	A6 8Ø 27 21		LB82D	LDA BEQ	,X+ LB852	GET INPUT CHAR BRANCH IF END OF LINE
3513	B831	ØD 43			TST	V43	* CHECK ILLEGAL TOKEN FLAG & BRANCH IF NOT
3515	B833	27 ØF			BEQ	LB844	* PROCESSING AN ILLEGAL TOKEN
3516	B835	BD B3 /	A2		JSR	LB3A2	SET CARRY IF NOT UPPER CASE ALPHA
3517	B838	24 18			BCC	LB852	BRANCH IF UPPER CASE ALPHA
3518 3519	B83A B83C	81 3Ø 25 Ø4			CMPA BLO	#'Ø LB842	* DON T CRUNCH ASCII NUMERIC CHARACTERS  * BRANCH IF NOT NUMERIC
3520	B83E	81 39			CMPA	#'9	*
3521	B84Ø	23 10			BLS	LB852	* BRANCH IF NUMERIC
3522	DO 4.2	ar 42			HERE IF CLR	NOT UPPER CASE ALPHA OR I	
3523 3524	B842 B844	ØF 43 81 20		LB842 LB844	CMPA	V43 #SPACE	CLEAR ILLEGAL TOKEN FLAG SPACE?
3525	B846	27 ØA			BEQ	LB852	DO NOT REMOVE SPACES
3526	B848	97 42			STA	V42	SAVE INPUT CHARACTER AS SCAN DELIMITER
3527 3528	B84A B84C	81 22 27 38			CMPA BEQ	#'" LB886	CHECK FOR STRING DELIMITER BRANCH IF STRING
3529	B84E	ØD 44			TST	V44	* CHECK DATA FLAG AND BRANCH IF CLEAR
3530	B85Ø	27 19			BEQ	LB86B	* DO NOT CRUNCH DATA
3531	B852	A7 CØ		LB852	STA	, U+	SAVE CHARACTER IN BUFFER
3532 3533	B854 B856	27 Ø6 81 3A			BEQ CMPA	LB85C #':	* CHECK FOR END OF SUBLINE
3534		27 CF			BEQ	LB829	* AND RESET FLAGS IF END OF SUBLINE
3535		20 D1		LB85A	BRA	LB82D	GO GET ANOTHER CHARACTER
3536		6F CØ		LB85C	CLR	, U+	* DOUBLE ZERO AT END OF LINE
3537 3538		6F CØ 1F 3Ø			CLR TFR	,U+ U,D	* SAVE ADDRESS OF END OF LINE IN ACCD
3539		83 Ø2 [	DA			#LINHDR	LENGTH OF LINE IN ACCD
3540		8E Ø2 [	DB		LDX	#LINBUF-1	* SET THE INPUT POINTER TO ONE BEFORE
3541		9F A6			STX	CHARAD	* THE START OF THE CRUNCHED LINE
3542 3543	B86A B86B	39 81 3F		LB86B	RTS CMPA	#'?	EXIT 'CRUNCH' CHECK FOR "?" - PRINT ABBREVIATION
3544	B86D	26 Ø4		LDOOD	BNE	LB873	BRANCH IF NOT PRINT ABBREVIATION
3545	B86F	86 87			LDA	#\$87	* GET THE PRINT TOKEN AND SAVE IT
3546		2Ø DF			BRA	LB852	* IN BUFFER
3547 3548		81 27 26 13		LB873	CMPA BNE	#'' LB88A	APOSTROPHE IS SAME AS REM BRANCH IF NOT REMARK
3549		CC 3A 8	83		LDD	#\$3A83	COLON, REM TOKEN
3550	B87A	ED C1			STD	,U++	SAVE IN BUFFER
3551		ØF 42		LB87C	CLR	V42	SET DELIMITER = Ø (END OF LINE)
3552 3553		A6 8Ø 27 DØ		LB87E	LDA BEQ	,X+ LB852	SCAN TILL WE MATCH [V42] BRANCH IF END OF LINE
3554		91 42				V42	DELIMITER?
3555		27 CC			BEQ	LB852	BRANCH OUT IF SO
3556		A7 CØ		LB886	STA	,U+	DON T CRUNCH REMARKS OR STRINGS
3557		20 F4		1 2004	BRA CMPA	LB87E	GO GET MORE STRING OR REMARK
3558 3559		81 3Ø 25 Ø4		LB88A	BCS	# W LB892	* LESS THAN ASCII ZERO? * BRANCH IF SO
3560		81 3C				#';+1	= CHECK FOR NUMERIC VALUE, COLON OR SEMICOLON

3561	B89Ø	25 0	α			BCS	LB852	= AND INSERT IN BUFFER IF SO
3562	B892	30 1			LB892		-1,X	MOVE INPUT POINTER BACK ONE
3563	B894	34 5			20032		U,X	SAVE POINTERS TO INPUT STRING, OUTPUT STRING
3564	B896	ØF 4	1			CLR	V41	TOKEN FLAG Ø = COMMAND, FF = SECONDARY
3565	B898	CE Ø		16		LDU	#COMVEC-1Ø	POINT U TO COMMAND INTERPRETATION
3566					*			TABLE FOR BASIC - 10
3567	B89B	ØF 4	12		LB89B	CLR	V42	INITIALIZE V42 AS TOKEN COUNTER
3568	B89D	33 4	ŀΑ		LB89D	LEAU	10,U	MOVE TO NEXT COMMAND INTERPRETATION TABLE
3569	B89F	A6 C	24			LDA	,U	GET NUMBER OF COMMANDS
357Ø	B8A1	27 3	31			BEQ	LB8D4	GO DO SECONDARY FUNCTIONS IF NO COMMAND TABLE
3571	B8A3	10 A	١E	41		LDY	1,U	POINT Y TO COMMAND DICTIONARY TABLE
3572	B8A6	AE E	4		LB8A6	LDX	,S	GET POINTER TO INPUT STRING
3573	B8A8	E6 A	١Ø		LB8A8	LDB	, Y+	GET A BYTE FROM DICTIONARY TABLE
3574	B8AA	EØ 8	8Ø			SUBB	, X+	SUBTRACT INPUT CHARACTER
3575	B8AC	27 F	Α			BEQ	LB8A8	LOOP IF SAME
3576	B8AE	C1 8	8Ø			CMPB	#\$80	LAST CHAR IN RESERVED WORD TABLE HAD
3577					*			BIT 7 SET, SO IF WE HAVE \$80 HERE
3578					*			THEN IT IS A GOOD COMPARE
3579	B8BØ	26 3	88			BNE	LB8EA	BRANCH IF NO MATCH - CHECK ANOTHER COMMAND
3580	B8B2	32 6				LEAS	2,\$	DELETE OLD INPUT POINTER FROM STACK
3581	B8B4	35 4				PULS	U	GET POINTER TO OUTPUT STRING
3582	B8B6	DA 4	12			ORB	V42	OR IN THE TABLE POSITION TO MAKE THE TOKEN
3583					*			- NOTE THAT B ALREADY HAD \$80 IN IT -
3584	B8B8	96 4				LDA	V41	* CHECK TOKEN FLAG AND BRANCH
3585	B8BA	26 0				BNE	LB8C2	* IF SECONDARY
3586		C1 8				CMPB		IS IT ELSE TOKEN?
3587	B8BE					BNE	LB8C6	NO
3588	B8CØ	86 3			1,0000	LDA	#':	PUT A COLON (SUBLINE) BEFORE ELSE TOKEN
3589		ED C			LB8C2	STD	, U++	SECONDARY TOKENS PRECEEDED BY \$FF
3590	B8C4	20 9			1,0006	BRA	LB85A	GO PROCESS MORE INPUT CHARACTERS
3591		E7 C			LB8C6	STB	, U+	SAVE THIS TOKEN
3592 3593	B8C8 B8CA	C1 8				CMPB BNE	LB8CE	DATA TOKEN? NO
3593		ØC 4				INC	V44	SET DATA FLAG
3595	B8CE	C1 8			LB8CE		#\$82	REM TOKEN?
3596	BBDØ	27 A			LBOCL	BEQ	LB87C	YES
3597		20 8			LB8D2	BRA	LB85A	GO PROCESS MORE INPUT CHARACTERS
3598	DODZ	20 0	, ,				CONDARY TOKEN	do PROCESS MORE INFOT CHARACTERS
3599	B8D4	CE Ø	11	1 R	LB8D4	LDU	#COMVEC-5	NOW DO SECONDARY FUNCTIONS
3600	B8D7	Ø3 4			2505.	COM	V41	TOGGLE THE TOKEN FLAG
3601	B8D9	26 0				BNE	LB89B	BRANCH IF NOW CHECKING SECONDARY COMMANDS
3602								
3603					* THIS CO	DE WILL	PROCESS INPUT DATA WHICH	CANNOT BE CRUNCHED AND SO
3604					* IS ASSU	MED TO	BE ILLEGAL DATA OR AN ILLE	GAL TOKEN
3605	B8DB	35 5	Ø			PULS	X,U	RESTORE INPUT AND OUTPUT POINTERS
3606	B8DD	A6 8	8Ø			LDA	, X+	* MOVE THE FIRST CHARACTER OF AN
3607	B8DF	A7 C	Ø			STA	,U+	* ILLEGAL TOKEN
3608	B8E1	BD B	33	A2		JSR	LB3A2	SET CARRY IF NOT ALPHA
3609	B8E4	25 E	С			BCS	LB8D2	BRANCH IF NOT ALPHA
3610	B8E6	Ø3 4	13			COM	V43	SET ILLEGAL TOKEN FLAG IF UPPER CASE ALPHA
3611	B8E8	2Ø E	8			BRA	LB8D2	PROCESS MORE INPUT CHARACTERS
3612	B8EA		12		LB8EA	INC	V42	INCREMENT TOKEN COUNTER
3613	B8EC	4 A				DECA		DECR COMMAND COUNTER
3614		27 A				BEQ	LB89D	GET ANOTHER COMMAND TABLE IF DONE W/THIS ONE
3615	B8EF	31 3					-1,Y	MOVE POINTER BACK ONE
3616		E6 A			LB8F1	LDB	, Y+	* GET TO NEXT
3617	B8F3	2A F				BPL	LB8F1	* RESERVED WORD
3618	B8F5	2Ø A	<b>\</b> F			BRA	LB8A6	GO SEE IF THIS WORD IS A MATCH
3619					+ DDINT			
3620	B8F7	27 5			* PRINT	DEO	1,000	DDANCH IE NO ADCHMENT
3621	B8F9				PRINT	BEQ BSR	LB958 LB8FE	BRANCH IF NO ARGUMENT CHECK FOR ALL PRINT OPTIONS
3622 3623	B8FB					CLR	DEVNUM	SET DEVICE NUMBER TO SCREEN
3624	B8FD		. 1			RTS	52711011	SE. SETTOE NOTIDER TO SURLER
3625	B8FE		ıø		LB8FE	CMPA	#'@	CHECK FOR PRINT @
3626	B9ØØ				250.2		LB907	NOT PRINT @
3627	B9Ø2			54			LA554	MOVE CURSOR TO PROPER PRINT LOCATION
3628	B9Ø5						LB911	GO PRINT THE DATA
3629	B9Ø7				LB9Ø7	CMPA	#'#	CHECK FOR PRINT NUMBER
3630	B9Ø9	26 0	3D			BNE	LB918	NOT PRINT#
3631	B9ØB			A5			LA5A5	CHECK FOR A VALID DEVICE NUMBER
3632	B9ØE	BD A	4	Ø6			LA406	CHECK FOR A VALID OUTPUT FILE
3633	B911	9D A	۱5		LB911		GETCCH	GET CURRENT INPUT CHARACTER
3634	B913	27 4	13			BEQ	LB958	BRANCH IF END OF LINE
3635	B915	BD B	32	6D		JSR	LB26D	SYNTAX CHECK FOR COMMA
3636	B918	BD Ø	31	79	LB918	JSR	RVEC9	HOOK INTO RAM
3637	B91B	27 4	18		LB91B	BEQ	LB965	RETURN IF END OF LINE
3638	B91D	81 A	١4		LB91D	CMPA	#\$A4	TOKEN FOR TAB( ?
3639	B91F					BEQ	LB97E	YES
3640	B921					CMPA		COMMA?
3641	B923					BEQ	LB966	YES - ADVANCE TO NEXT TAB FIELD
3642	B925					CMPA		SEMICOLON?
3643	B927						LB997	YES - DO NOT ADVANCE CURSOR
3644	B929			56		JSR	LB156	EVALUATE EXPRESSION
3645	B92C					LDA	VALTYP	* GET VARIABLE TYPE AND
3646	B92E					PSHS		* SAVE IT ON THE STACK
3647	B93Ø			0.0			LB938	BRANCH IF STRING VARIABLE
3648	B932						LBDD9	CONVERT FP NUMBER TO AN ASCII STRING
3649	B935	RD E	55	10		JSR	LB516	PARSE A STRING FROM (X-1) AND PUT

3650						*			DESCRIPTOR ON STRING STACK
3651	B938	8D	65			LB938	BSR	LB99F	PRINT STRING POINTED TO BY X
3652	B93A	35				25300	PULS	В	GET VARIABLE TYPE BACK
3653	B93C		A3	5F			JSR	LA35F	SET UP TAB WIDTH ZONE, ETC
3654	B93F	ØD	6 E				TST	PRTDEV	* CHECK THE PRINT DEVICE
3655	B941	27	Ø6				BEQ	LB949	* AND BRANCH IF NOT CASSETTE
3656	B943	8D	13				BSR	LB958	SEND A CARRIAGE RETURN TO CONSOLE OUT
3657	B945	9 D	Α5				JSR	GETCCH	GET CURRENT INPUT CHARACTER
3658	B947	20	D2				BRA	LB91B	CHECK FOR MORE PRINT DATA
3659	B949	5 D				LB949	TSTB		CHECK CURRENT PRINT POSITION
3660	B94A	26	Ø8				BNE	LB954	BRANCH IF NOT AT START OF LINE
3661	B94C	9 D	Α5				JSR	GETCCH	GET CURRENT INPUT CHARACTER
3662	B94E	81	20				CMPA	#',	COMMA?
3663	B95Ø	27	14				BEQ	LB966	SKIP TO NEXT TAB FIELD
3664	B952	8D					BSR	LB9AC	SEND A SPACE TO CONSOLE OUT
3665	B954	9 D				LB954	JSR	GETCCH	GET CURRENT INPUT CHARACTER
3666	B956	26					BNE	LB91D	BRANCH IF NOT END OF LINE
3667	B958	86				LB958	LDA	#CR	* SEND A CR TO
3668	B95A	20					BRA	LB9B1	* CONSOLE OUT
3669	B95C		A3	5 F		LB95C	JSR	LA35F	SET UP TAB WIDTH, ZONE ETC
3670	B95F	27					BEQ	LB958	BRANCH IF WIDTH = ZERO
3671	B961	96					LDA	DEVPOS	GET PRINT POSITION
3672	B963	26	F3				BNE	LB958	BRANCH IF NOT AT START OF LINE
3673	B965	39				LB965	RTS	FAD	
3674	DOCC	D.D.	4.2			* SKIP TO			CET UD TAD HIDTH ZONE ETC
3675	B966	BD		31		LB966	JSR	LA35F	SET UP TAB WIDTH, ZONE ETC BRANCH IF LINE WIDTH = Ø (CASSETTE)
3676 3677	B969 B96B	27 D6					BEQ LDB	LB975 DEVPOS	GET CURRENT POSITION
3678	B96D	D1					CMPB	DEVLCF	COMPARE TO LAST TAB ZONE
3679	B96F	25					BCS	LB977	BRANCH IF < LAST TAB ZONE
3680	B971	8D					BSR	LB958	SEND A CARRIAGE RETURN TO CONSOLE OUT
3681	B973	20					BRA	LB997	GET MORE DATA
3682	B975	D6				LB975	LDB	DEVPOS	*
3683	B977	DØ				LB977	SUBB	DEVCFW	* SUBTRACT TAB FIELD WIDTH FROM CURRENT
3684	B979	24				200777	BCC	LB977	* POSITION UNTIL CARRY SET - NEGATING THE
3685	B97B	50					NEGB		* REMAINDER LEAVES THE NUMBER OF SPACES TO NEXT
3686						*			* TAB ZONE IN ACCB
3687	B97C	20	10				BRA	LB98E	GO ADVANCE TO NEXT TAB ZONE
3688									
3689						* PRINT T	AB(		
3690	B97E	BD	В7	Ø9		LB97E	JSR	LB7Ø9	EVALUATE EXPRESSION - RETURN VALUE IN B
3691	B981	81	29				CMPA	#')	* 'SYNTAX' ERROR IF NOT ')'
3692	B983	10	26	F8	FØ		LBNE	LB277	*
3693	B987	BD	А3	5 F			JSR	LA35F	SET UP TAB WIDTH, ZONE ETC
3694	B98A	DØ	6C				SUBB	DEVPOS	GET DIFFERENCE OF PRINT POSITION & TAB POSITION
3695	B98C	23					BLS	LB997	BRANCH IF TAB POSITION < CURRENT POSITION
3696	B98E	ØD				LB98E	TST	PRTDEV	* GET PRINT DEVICE NUMBER AND
3697	B99Ø	26					BNE	LB997	* BRANCH IF CASSETTE
3698	B992	8D	18			LB992	BSR	LB9AC	SEND A SPACE TO CONSOLE OUT
3699	B994	5 A					DECB		DECREMENT DIFFERENCE COUNT
3700	B995	26					BNE	LB992	BRANCH UNTIL CURRENT POSITION = TAB POSITION
3701	B997	9 D				LB997	JSR	GETNCH	GET NEXT CHARACTER FROM BASIC
3702	B999	/ E	В9	18		+ CODY 4	JMP	LB91B	LOOK FOR MORE PRINT DATA
3703	DOOC	D.D.	DГ	10				FROM (X) TO CONSOLE OUT	PARSE A STRING FROM X AND PUT
3704	B99C	ви	В5	10		LB99C *	JSR	LB518	DESCRIPTOR ON STRING STACK
37Ø5 37Ø6	B99F	D D	D.C	E 7		LB99F	100	1.045.7	GET LENGTH OF STRING AND REMOVE
37Ø7	вээг	ви	В6	5/		LB99F	JSR	LB657	
3708	B9A2								
3700	DJAL	5.0				*	TNCR		DESCRIPTOR FROM STRING STACK
	B943	5 C				* LB9A3	INCB		DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW
371Ø	B9A3 B9A4	5C 5A 27	BF			LB9A3	DECB	LB965	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER
	B9A4	5A 27				LB9A3	DECB BEQ		DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE
3710 3711 3712		5A 27 A6	8Ø			LB9A3	DECB	LB965 ,X+ LB9B1	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER
3711	B9A4 B9A6	5A 27 A6 8D	8Ø Ø7			* LB9A3	DECB BEQ LDA BSR	, χ+	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X
3711 3712	B9A4 B9A6 B9A8	5A 27 A6 8D 2Ø	8Ø Ø7 F7			EB9A3	DECB BEQ LDA BSR	, X+ LB9B1 LB9A3	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT
3711 3712 3713	B9A4 B9A6 B9A8 B9AA	5A 27 A6 8D 2Ø 86	8Ø Ø7 F7 2Ø				DECB BEQ LDA BSR BRA	, X+ LB9B1 LB9A3	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING
3711 3712 3713 3714 3715 3716	B9A4 B9A6 B9A8 B9AA B9AC B9AE	5A 27 A6 8D 2Ø 86 8C 86	8Ø Ø7 F7 2Ø			LB9AC LB9AF	DECB BEQ LDA BSR BRA LDA FCB LDA	,X+ LB9B1 LB9A3 #SPACE SKP2 #'?	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717	B9A4 B9A6 B9A8 B9AA B9AC B9AE	5A 27 A6 8D 2Ø 86 8C 86	8Ø Ø7 F7 2Ø			LB9AC	DECB BEQ LDA BSR BRA LDA FCB LDA	,X+ LB9B1 LB9A3 #SPACE SKP2	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES
3711 3712 3713 3714 3715 3716 3717 3718	B9A4 B9A6 B9A8 B9AA B9AC B9AE	5A 27 A6 8D 2Ø 86 8C 86	8Ø Ø7 F7 2Ø			LB9AC LB9AF LB9B1	DECB BEQ LDA BSR BRA LDA FCB LDA JMP	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719	B9A4 B9A6 B9A8 B9AA B9AC B9AE	5A 27 A6 8D 2Ø 86 8C 86	8Ø Ø7 F7 2Ø			LB9AC LB9AF LB9B1	DECB BEQ LDA BSR BRA LDA FCB LDA JMP	,X+ LB9B1 LB9A3 #SPACE SKP2 #'?	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720	B9A4 B9A6 B9A8 B9AA B9AC B9AE	5A 27 A6 8D 2Ø 86 8C 86	8Ø Ø7 F7 2Ø			LB9AC LB9AF LB9B1 * FLOATIN	DECB BEQ LDA BSR BRA LDA FCB LDA JMP	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721	B9A4 B9A6 B9A8 B9AA B9AC B9AE B9AF B9B1	5A 27 A6 8D 2Ø 86 8C 86 7E	8Ø Ø7 F7 2Ø 3F A2	82		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR T MATH PACKAGE	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9AF B9B1	5A 27 A6 8D 2Ø 86 8C 86 7E	80 07 F7 20 3F A2	82 CØ		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  T MATH PACKAGE ####################################	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723	B9A4 B9A6 B9A8 B9AA B9AC B9AE B9AF B9B1	5A 27 A6 8D 2Ø 86 8C 86 7E	80 07 F7 20 3F A2	82 CØ		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAR LDX BRA	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  F MATH PACKAGE  ###################################	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9AF B9B1	5A 27 A6 8D 2Ø 86 8C 86 7E	80 07 F7 20 3F A2	82 CØ		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4 * SUBTRAC	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX BRA T FPAØ	,X+ LB9B1 HSPACE SKP2 #'? PUTCHR  MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9B1	5A 27 A6 8D 20 86 8C 86 7E	8Ø Ø7 F7 2Ø 3F A2 BE Ø9	82 CØ		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4 * SUBTRAC * TO BY (	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX BRA T FPAØ X), LEA	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  T MATH PACKAGE  # LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9AF B9B1	5A 27 A6 8D 20 86 8C 86 7E	8Ø Ø7 F7 2Ø 3F A2 BE Ø9	82 CØ		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4 * SUBTRAC	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX BRA T FPAØ X), LEA	,X+ LB9B1 HSPACE SKP2 #'? PUTCHR  MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9B1	5A 27 A6 8D 20 86 8C 86 7E	8Ø Ø7 F7 2Ø 3F A2 BE Ø9	82 CØ		LB9AC LB9AF LB9B1  * FLOATIN  * ADD .5 LB9B4  * SUBTRAC  * TO BY ( LB9B9	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX BRA T FPAØ X), LEA	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  「MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9B1	5A 27 A6 8D 20 86 8C 86 7E	8Ø Ø7 F7 2Ø 3F A2 BE Ø9	82 CØ		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4 * SUBTRAC * TO BY ( LB9B9 * ARITHME	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX BRA T FPAØ X), LEA JSR TIC OPE	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR I MATH PACKAGE  #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F ERATION (-) JUMPS HERE - SL	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1 UBTRACT FPAØ FROM FPA1 (ENTER
3711 3712 3713 3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9B1	5A 27 A6 8D 2Ø 86 8C 86 7E	80 07 F7 20 3F A2 BE 09	82 CØ		LB9AC LB9AF LB9B1  * FLOATIN  * ADD .5 LB9B4  * SUBTRAC  * TO BY ( LB9B9  * ARITHME  * WITH EX	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX BRA T FPAØ X), LEA JSR TIC OPE	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  T MATH PACKAGE  #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F ERATION (-) JUMPS HERE - SI OF FPAØ IN ACCB AND EXPONE	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA)
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3727 3728	B9A4 B9A6 B9A8 B9AC B9AE B9AF B9B1	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	80 07 F7 20 3F A2 BE 09 BB	82 CØ 2F		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4 * SUBTRAC * TO BY ( LB9B9 * ARITHME	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAG LDX BRA T FPAG X), LEA JSR TIC OPE PONENT COM	,X+ LB9B1 HSPACE SKP2 #'? PUTCHR    MATH PACKAGE   HLBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F  ERATION (-) JUMPS HERE - SL OF FPAØ IN ACCB AND EXPONE FPØSGN	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEPP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA) CHANGE MANTISSA SIGN OF FPAØ
3711 3712 3713 3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729	B9A4 B9A6 B9A8 B9AA B9AC B9AF B9B1	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	8Ø Ø7 F7 2Ø 3F A2 BE Ø9 BB	82 CØ 2F		LB9AC LB9AF LB9B1  * FLOATIN  * ADD .5 LB9B4  * SUBTRAC  * TO BY ( LB9B9  * ARITHME  * WITH EX	DECB BEQ LDA BSR BRA LDA JMP G POINT TO FPAØ LDX BRA T FPAØ X), LEA JSR TIC OPE PONENT COM COM	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  T MATH PACKAGE  #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F ERATION (-) JUMPS HERE - SI OF FPAØ IN ACCB AND EXPONE	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA)
3711 3712 3713 3714 3715 3716 3717 3718 3729 3721 3722 3723 3724 3725 3726 3727 3728 3727 3728 3729	B9A4 B9A6 B9A8 B9AC B9AE B9AF B9B1 B9B4 B9B7	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	8Ø Ø7 F7 2Ø 3F A2 BE Ø9 BB	82 CØ 2F		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4 * SUBTRAC * TO BY ( LB9B9 * ARITHME * WITH EX LB9BC	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINT TO FPAØ LDX BRA T FPAØ X), LEA JSR TIC OPE PONENT COM COM BRA	,X+ LB9B1 #SPACE SKP2 #'? PUTCHR  MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F  ERATION (-) JUMPS HERE - SL OF FPAØ IN ACCB AND EXPONE FPØSGN RESSGN	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA) CHANGE MANTISSA SIGN OF FPAØ REVERSE RESULT SIGN FLAG
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3731	B9A4 B9A6 B9A8 B9AC B9AE B9AF B9B1 B9B4 B9B7	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	8Ø Ø7 F7 2Ø 3F A2 BE Ø9 BB	82 CØ 2F		LB9AC LB9AF LB9B1  * FLOATIN  * ADD .5 LB9B4  * SUBTRAC  * TO BY ( LB9B9  * ARITHME  * WITH EX LB9BC  * ADD FP	DECB BEQ LDA BSR BRA LDA JMP G POINT TO FPAØ LDX BRA T FPAØ JSR TIC OPE PONENT COM COM BRA NUMBER	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  I MATH PACKAGE  # HLBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F ERATION (-) JUMPS HERE - SL OF FPAØ IN ACCB AND EXPONE FPØSGN LB9C5 LB9C5	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA) CHANGE MANTISSA SIGN OF FPAØ REVERSE RESULT SIGN FLAG
3711 3712 3713 3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733	B9A4 B9A6 B9A8 B9AC B9AE B9AF B9B1 B9B4 B9B7	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	8Ø Ø7 F7 2Ø 3F A2 BE Ø9 BB	82 CØ 2F		LB9AC LB9AF LB9B1  * FLOATIN  * ADD .5 LB9B4  * SUBTRAC  * TO BY ( LB9B9  * ARITHME  * WITH EX LB9BC  * ADD FP  * (X) TO	DECB BEQ LDA BSR BRA LDA JMP G POINT TO FPAØ LDX BRA T FPAØ JSR TIC OPE PONENT COM COM BRA NUMBER	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  「MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F  ERATION (-) JUMPS HERE - SL OF FPAØ IN ACCB AND EXPONE FPØSGN RESSGN LB9C5 POINTED TO BY LEAVE RESULT IN FPAØ	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA) CHANGE MANTISSA SIGN OF FPAØ REVERSE RESULT SIGN FLAG
3711 3712 3713 3714 3715 3716 3717 3718 3720 3721 3722 3723 3724 3725 3726 3727 3728 3727 3738 3730 3731 3732 3733	B9A4 B9A6 B9A8 B9AC B9AE B9AF B9B1 B9B4 B9B7 B9B8 B9B9	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	8Ø Ø7 F7 2Ø 3F A2 BE Ø9 BB	82 CØ 2F		LB9AC LB9AF LB9B1  * FLOATIN  * ADD .5 LB9B4  * SUBTRAC  * TO BY ( LB9B9  * ARITHME  * WITH EX LB9BC  * ADD FP  * (X) TO	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINTTO FPA@ LDX BRA LDX BRA LDX BRA LDX LDX BRA LDX LDX COM COM BRA BRA BRA LDX	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  「MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F  ERATION (-) JUMPS HERE - SL OF FPAØ IN ACCB AND EXPONE FPØSGN RESSGN LB9C5 POINTED TO BY LEAVE RESULT IN FPAØ	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEPP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA) CHANGE MANTISSA SIGN OF FPAØ REVERSE RESULT SIGN FLAG GO ADD FPA1 AND FPAØ
3711 3712 3713 3714 3715 3716 3717 3718 3729 3721 3722 3723 3724 3725 3726 3727 3728 3729 3739 3731 3732 3733 3734 3733	B9A4 B9A6 B9A8 B9AC B9AE B9AF B9B1 B9B4 B9B7 B9B8 B9B9	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	8Ø Ø7 F7 2Ø 3F A2 BE Ø9 BB	82 CØ 2F		LB9AC LB9AF LB9B1 * FLOATIN * ADD .5 LB9B4 * SUBTRAC * TO BY ( LB9B9 * ARITHME * WITH EX LB9BC  * ADD FP * (X) TO LB9C2	DECB BEQ LDA BSR BRA LDA FCB LDA JMP G POINTTO FPA@ LDX BRA LDX BRA LDX BRA LDX LDX BRA LDX LDX COM COM BRA BRA BRA LDX	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  「MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F  ERATION (-) JUMPS HERE - SL OF FPAØ IN ACCB AND EXPONE FPØSGN RESSGN LB9C5 POINTED TO BY LEAVE RESULT IN FPAØ	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEEP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1  UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA) CHANGE MANTISSA SIGN OF FPAØ REVERSE RESULT SIGN FLAG GO ADD FPA1 AND FPAØ  UNPACK PACKED FP DATA FROM (X) TO
3711 3712 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728 3729 3730 3731 3732 3733 3734 3735	B9A4 B9A6 B9A8 B9AC B9AE B9AF B9B1 B9B4 B9B7 B9B8 B9B9	5A 27 A6 8D 20 86 8C 86 7E 8E 20 8D	8Ø Ø7 F7 2Ø 3F A2 BE Ø9 BB	82 CØ 2F		LB9AC LB9AF LB9B1  * FLOATIN  * ADD .5 LB9B4  * SUBTRAC  * TO BY ( LB9B9  * ARITHME  * WITH EX LB9BC  * ADD FP  * (X) TO LB9C2  *	DECB BEQ LDA BSR BRA LDA LDA JMP G POINT TO FPA® LDX BRA LDX BRA JT FPA® LDX BRA T FPA® COM COM COM COM COM SOM BRA BRA JSR FPA® FPA® FPA® JSR	,X+ LB9B1 LB9A3 #SPACE SKP2 #'? PUTCHR  「MATH PACKAGE  # #LBECØ LB9C2 FROM FP NUMBER POINTED AVE RESULT IN FPAØ LBB2F  ERATION (-) JUMPS HERE - SL OF FPAØ IN ACCB AND EXPONE FPØSGN RESSGN LB9C5 POINTED TO BY LEAVE RESULT IN FPAØ	DESCRIPTOR FROM STRING STACK COMPENSATE FOR DECB BELOW DECREMENT COUNTER EXIT ROUTINE GET A CHARACTER FROM X SEND TO CONSOLE OUT KEPP LOOPING SPACE TO CONSOLE OUT SKIP NEXT TWO BYTES QUESTION MARK TO CONSOLE OUT JUMP TO CONSOLE OUT  FLOATING POINT CONSTANT (.5) ADD .5 TO FPAØ  COPY PACKED FP DATA FROM (X) TO FPA1 UBTRACT FPAØ FROM FPA1 (ENTER ENT OF FPA1 IN ACCA) CHANGE MANTISSA SIGN OF FPAØ REVERSE RESULT SIGN FLAG GO ADD FPA1 AND FPAØ

3739			* FPA1 (	ENTER WI	TH EXPONENT OF FPAØ IN	ACCB AND EXPONENT OF FPA1 IN ACCA
3740	B9C5	5 D	LB9C5	TSTB		CHECK EXPONENT OF FPAØ
3741	B9C6	10 27 02 8	Ø		LBC4A	COPY FPA1 TO FPAØ IF FPAØ = Ø
3742	B9CA	8E ØØ 5C		LDX	#FP1EXP	POINT X TO FPA1
3743	B9CD B9CF	1F 89	LB9CD	TFR	A,B	PUT EXPONENT OF FPA1 INTO ACCB
3744 3745	B9DØ	5D 27 6C		TSTB BEQ	LBA3E	CHECK EXPONENT  RETURN IF EXPONENT = Ø (ADDING Ø TO FPAØ)
3746		DØ 4F			FPØEXP	SUBTRACT EXPONENT OF FPAØ FROM EXPONENT OF FPA1
3747	B9D4	27 69		BEQ	LBA3F	BRANCH IF EXPONENTS ARE EQUAL
3748	B9D6	25 ØA		BCS	LB9E2	BRANCH IF EXPONENT FPAØ > FPA1
3749	B9D8	97 4F		STA	FPØEXP	REPLACE FPAØ EXPONENT WITH FPA1 EXPONENT
3750	B9DA	96 61		LDA	FP1SGN	* REPLACE FPAØ MANTISSA SIGN
3751	B9DC	97 54		STA	FPØSGN	* WITH FPA1 MANTISSA SIGN
3752		8E ØØ 4F		LDX	#FPØEXP	POINT X TO FPAØ
3753 3754	B9E1	5Ø C1 F8	LB9E2	NEGB CMPB	#- Q	NEGATE DIFFERENCE OF EXPONENTS TEST DIFFERENCE OF EXPONENTS
3755		2F 59	LD9E2	BLE	LBA3F	BRANCH IF DIFFERENCE OF EXPONENTS <= 8
3756	B9E6			CLRA	EBROI	CLEAR OVERFLOW BYTE
3757	B9E7	64 Ø1		LSR	1,X	SHIFT MS BYTE OF MANTISSA; BIT 7 = Ø
3758	B9E9	BD BA BA		JSR	LBABA	GO SHIFT MANTISSA OF (X) TO THE RIGHT (B) TIMES
3759	B9EC	D6 62	LB9EC	LDB	RESSGN	GET SIGN FLAG
3760		2A ØB			LB9FB	BRANCH IF FPAØ AND FPA1 SIGNS ARE THE SAME
3761		63 Ø1			1,X	* COMPLEMENT MANTISSA POINTED
3762		63 Ø2		COM	2,X	* TO BY (X) THE
3763 3764	B9F4 B9F6	63 Ø3 63 Ø4		COM COM	3,X 4,X	* ADCA BELOW WILL * CONVERT THIS OPERATION
3765	B9F8	43		COMA	т, к	* INTO A NEG (MANTISSA)
3766	B9F9			ADCA	#0	ADD ONE TO ACCA - COMA ALWAYS SETS THE CARRY FLAG
3767			* THE PR		WO BYTES MAY BE REPLAC	
3768			*			
3769			* ADD MA	NTISSAS	OF FPAØ AND FPA1, PUT	RESULT IN FPAØ
3770		97 63	LB9FB	STA	FPSBYT	SAVE FPA SUB BYTE
3771		96 53		LDA	FPAØ+3	* ADD LS BYTE
3772	B9FF	99 60			FPA1+3	* OF MANTISSA
3773 3774	BAØ1	97 53 96 52		STA LDA	FPAØ+3	SAVE IN FPAØ LSB
3775	BAØ5	90 52 99 5F			FPAØ+2 FPA1+2	* ADD NEXT BYTE * OF MANTISSA
3776	BAØ7	97 52		STA	FPAØ+2	SAVE IN FPAØ
3777	BAØ9	96 51		LDA	FPAØ+1	* ADD NEXT BYTE
3778	BAØB	99 5E		ADCA	FPA1+1	* OF MANTISSA
3779	BAØD	97 51		STA	FPAØ+1	SAVE IN FPAØ
3780	BAØF	96 5Ø		LDA	FPAØ	* ADD MS BYTE
3781		99 5D		ADCA		* OF MANTISSA
3782		97 50		STA	FPAØ	SAVE IN FPAØ
3783	BA15	5D		TSTB	LDAFC	TEST SIGN FLAG
3784 3785		2A 44 25 Ø2	LBA18	BPL BCS	LBA5C LBA1C	BRANCH IF FPAØ & FPA1 SIGNS WERE ALIKE BRANCH IF POSITIVE MANTISSA
3786		8D 5D	LDAIO	BSR	LBA79	NEGATE FPAØ MANTISSA
3787	5/12/1	05 05		5011	25117 5	REALIZE TITLE THAT TOOK
3788			* NORMAL	IZE FPAG	1	
3789	BA1C	5 F	LBA1C	CLRB		CLEAR TEMPORARY EXPONENT ACCUMULATOR
3790	BA1D	96 5Ø	LBA1D	LDA	FPAØ	TEST MSB OF MANTISSA
3791	BA1F	26 2E			LBA4F	BRANCH IF <> Ø
3792	BA21	96 51			FPAØ+1	* IF THE MSB IS
3793 3794	BA23	97 5Ø 96 52		STA LDA	FPAØ FPAØ+2	* Ø, THEN SHIFT THE
3795		97 51			FPAØ+1	* MANTISSA A WHOLE BYTE * AT A TIME. THIS
3796		96 53			FPAØ+3	* IS FASTER THAN ONE
3797		97 52			FPAØ+2	* BIT AT A TIME
3798	BA2D	96 63		LDA	FPSBYT	* BUT USES MORE MEMORY.
3799	BA2F	97 53		STA	FPAØ+3	* FPSBYT, THE CARRY IN
3800		ØF 63		CLR	FPSBYT	
38Ø1	BA33	CD GO				* BYTE, REPLACES THE MATISSA LSB.
2000				ADDB	#8	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT
3802	BA35	C1 28		ADDB CMPB	#8 #5*8	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS
38Ø3	BA37	C1 28 2D E4	I PAZO	ADDB CMPB BLT	#8	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø
38Ø3 38Ø4	BA37 BA39	C1 28 2D E4 4F	LBA39	ADDB CMPB BLT CLRA	#8 #5*8 LBA1D	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = 0 A ZERO EXPONENT = 0 FLOATING POINT
38Ø3 38Ø4 38Ø5	BA37 BA39 BA3A	C1 28 2D E4 4F 97 4F	LBA39 LBA3A	ADDB CMPB BLT CLRA STA	#8 #5*8 LBA1D FPØEXP	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT
38Ø3 38Ø4 38Ø5 38Ø6	BA37 BA39 BA3A BA3C	C1 28 2D E4 4F	LBA3A	ADDB CMPB BLT CLRA STA STA	#8 #5*8 LBA1D	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = 0 A ZERO EXPONENT = 0 FLOATING POINT
38Ø3 38Ø4 38Ø5	BA37 BA39 BA3A	C1 28 2D E4 4F 97 4F 97 54		ADDB CMPB BLT CLRA STA	#8 #5*8 LBA1D FPØEXP	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT
3803 3804 3805 3806 3807	BA37 BA39 BA3A BA3C BA3E	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D	LBA3A LBA3E	ADDB CMPB BLT CLRA STA STA RTS	#8 #5*8 LBAID FPØEXP FPØSGN	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN
38Ø3 38Ø4 38Ø5 38Ø6 38Ø7 38Ø8	BA37 BA39 BA3A BA3C BA3E BA3F BA41	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D	LBA3A LBA3E LBA3F	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA	#8 #5*8 LBAID FPØEXP FPØSGN LBAAE LB9EC	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPAØ MANTISSA TO RIGHT
3803 3804 3805 3806 3807 3808 3809 3810 3811	BA37 BA39 BA3A BA3C BA3E BA3F BA41	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F	LBA3A LBA3E LBA3F * SHIFT	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEF	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC T ONE BIT UNTIL BIT 7	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPAØ MANTISSA TO RIGHT
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8	LBA3A LBA3E LBA3F  * SHIFT * OF MAT	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEF	#8 #5*8 LBAID FPØEXP FPØSGN LBAAE LB9EC	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8	LBA3A LBA3E LBA3F * SHIFT	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEF	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8	LBA3A LBA3E LBA3F  * SHIFT * OF MAT	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEFI ISSA MS INCB ASL	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814 3815	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8	LBA3A LBA3E LBA3F  * SHIFT * OF MAT	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEF	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPAØ+3	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = 0 A ZERO EXPONENT = 0 FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPA0 MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT LS BYTE
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8	LBA3A LBA3E LBA3F  * SHIFT * OF MAT	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEFI ISSA MS INCB ASL ROL	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814 3815 3816	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42 BA44 BA45 BA47 BA49	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8 5C Ø8 63 Ø9 53 Ø9 52	LBA3A LBA3E LBA3F  * SHIFT * OF MAT	ADDB CMPB BLT CLRA STA RTS BSR CLRB BRA FPAØ LEF ISSA MS INCB ASL ROL ROL	#8 #5*8 LBA1D  FPØEXP FPØSGN  LBAAE  LB9EC TO ONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPAØ+3 FPAØ+2	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT LS BYTE SHIFT NS BYTE
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814 3815 3816 3817	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42 BA44 BA45 BA47 BA49 BA4B	C1 28 2D E4 4F 97 4F 97 54 39 6D 5F 2Ø A8 5C 08 63 09 53 09 52 09 51	LBA3A LBA3E LBA3F  * SHIFT * OF MAT	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEI ISSA MS INCB ASL ROL ROL ROL	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPA0+3 FPA0+2 FPA0+1	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42 BA44 BA45 BA47 BA49 BA4B BA4D BA4F	C1 28 2D E4 4F 97 4F 97 54 39 6D 5F 2Ø A8 5C 08 63 09 53 09 52 09 51 09 50 2A F3 96 4F	LBA3A  LBA3E  LBA3F  * SHIFT  * OF MAT  LBA44	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FPAØ LEF ISSA MS INCB ASL ROL ROL ROL ROL ROL BPL LDA	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPA0+3 FPA0+2 FPA0+1 FPA0 LBA44 FPØEXP	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT MS BYTE SHIFT MS BYTE BRANCH IF NOT YET NORMALIZED GET CURRENT EXPONENT
3803 3804 3805 3806 3807 3808 3809 3811 3812 3813 3814 3815 3816 3817 3818 3819 3818 3819	BA37 BA39 BA3A BA3C BA3E BA41 BA42 BA44 BA45 BA47 BA49 BA4B BA4D BA4F BA51 BA53	C1 28 2D E4 4F 97 4F 97 54 39 6D 5F 2Ø A8 5C 08 63 09 53 09 52 09 51 09 51 09 51 09 54 34 04	LBA3A  LBA3E  LBA3F  * SHIFT  * OF MAT  LBA44	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA EFPAØ LEFI ISSA MS INCB ASL ROL ROL ROL BPL LDA PSHS	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPA0+3 FPA0+2 FPA0+2 FPA0+1 FPA0 LBA44 FPA0 LBA44 FPØEXP B	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT LS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT MS BYTE SHIFT MS BYTE BRANCH IF NOT YET NORMALIZED GET CURRENT EXPONENT SAVE EXPONENT MODIFIER CAUSED BY NORMALIZATION
3803 3804 3805 3806 3807 3808 3810 3811 3812 3813 3814 3815 3816 3816 3817 3818 3819 3821 3822	BA37 BA39 BA3A BA3C BA3E BA41 BA42 BA44 BA45 BA47 BA49 BA4B BA4D BA53 BA53	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8 5C 08 63 09 53 09 52 09 51 09 50 2A F3 96 4F 34 04 AØ EØ	LBA3A  LBA3E  LBA3F  * SHIFT  * OF MAT  LBA44	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FFPAØ LES INCB ASL ROL ROL ROL ROL BPL LDA PSHS SUBA	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPAØ+3 FPAØ+2 FPAØ+1 FPAØ LBA44 FPØEXP B B ,S+	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = 0 A ZERO EXPONENT = 0 FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE BRANCH IF NOT YET NORMALIZED GET CURRENT EXPONENT SAVE EXPONENT MODIFIER CAUSED BY NORMALIZATION SUBTRACT ACCUMULATED EXPONENT MODIFIER
3803 3804 3805 3806 3807 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42 BA44 BA45 BA47 BA49 BA4F BA51 BA51 BA55 BA55 BA55	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8 5C Ø8 63 Ø9 53 Ø9 52 Ø9 51 Ø9 50 2A F3 96 4F 34 Ø4 AØ 69 97 4F	LBA3A  LBA3E  LBA3F  * SHIFT  * OF MAT  LBA44	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA EFRA EFRA INCB ASL ROL ROL ROL ROL ROL ROL ROL SPL LDA PSHS SUBA STA	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPA0+3 FPA0+2 FPA0+1 FPA0 LBA44 FPØEXP  B ,S+ FPØEXP	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT NS BYTE BRANCH IF NOT YET NORMALIZED GET CURRENT EXPONENT SAVE EXPONENT MODIFIER CAUSED BY NORMALIZATION SUBTRACT ACCUMULATED EXPONENT MODIFIER SAVE AS NEW EXPONENT
3803 3804 3805 3806 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42 BA44 BA45 BA47 BA49 BA4F BA51 BA51 BA55 BA55 BA55	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8 5C 08 63 09 53 09 52 09 51 09 50 2A F3 96 4F 34 04 AØ EØ	LBA3A  LBA3E  LBA3F  * SHIFT  * OF MAT  LBA44	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA FFPAØ LES INCB ASL ROL ROL ROL ROL BPL LDA PSHS SUBA	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPAØ+3 FPAØ+2 FPAØ+1 FPAØ LBA44 FPØEXP B B ,S+	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT LS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT MS BYTE BRANCH IF NOT YET NORMALIZED GET CURRENT EXPONENT SAVE EXPONENT MODIFIER CAUSED BY NORMALIZATION SUBTRACT ACCUMULATED EXPONENT SAVE AS NEW EXPONENT SET FPAØ = Ø IF THE NORMALIZATION CAUSED
3803 3804 3805 3806 3807 3808 3809 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42 BA44 BA45 BA47 BA49 BA4F BA51 BA51 BA55 BA55 BA55	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8 5C Ø8 63 Ø9 53 Ø9 52 Ø9 51 Ø9 50 2A F3 96 4F 34 Ø4 AØ 69 97 4F	LBA3A  LBA3E  LBA3F  * SHIFT  * OF MAT  LBA44	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA EFRA EFRA INCB ASL ROL ROL ROL ROL ROL ROL ROL SPL LDA PSHS SUBA STA	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPA0+3 FPA0+2 FPA0+1 FPA0 LBA44 FPØEXP  B ,S+ FPØEXP	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = 0 A ZERO EXPONENT = 0 FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPA0 MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT SUB BYTE ONE LEFT SHIFT LS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE BRANCH IF NOT YET NORMALIZED GET CURRENT EXPONENT SAVE EXPONENT MODIFIER CAUSED BY NORMALIZATION SUBTRACT ACCUMULATED EXPONENT MODIFIER SAVE AS NEW EXPONENT SET FPA0 = 0 IF THE NORMALIZATION CAUSED MORE OR EQUAL NUMBER OF LEFT SHIFTS THAN THE
3803 3804 3805 3806 3807 3808 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824	BA37 BA39 BA3A BA3C BA3E BA3F BA41 BA42 BA44 BA45 BA47 BA49 BA4F BA51 BA51 BA55 BA55 BA55	C1 28 2D E4 4F 97 4F 97 54 39 8D 6D 5F 2Ø A8 5C Ø8 63 Ø9 53 Ø9 52 Ø9 51 Ø9 50 2A F3 96 4F 34 Ø4 AØ EØ 97 4F 23 DE	LBA3A  LBA3E  LBA3F  * SHIFT  * OF MAT  LBA44	ADDB CMPB BLT CLRA STA STA RTS BSR CLRB BRA EFRA EFRA INCB ASL ROL ROL ROL ROL ROL ROL ROL SPL LDA PSHS SUBA STA	#8 #5*8 LBAID  FPØEXP FPØSGN  LBAAE  LB9EC TONE BIT UNTIL BIT 7 BYTE = 1  FPSBYT FPA0+3 FPA0+2 FPA0+1 FPA0 LBA44 FPØEXP  B ,S+ FPØEXP	SHIFTING ONE BYTE = 8 BIT SHIFTS; ADD 8 TO EXPONENT CHECK FOR 5 SHIFTS BRANCH IF < 5 SHIFTS, IF > 5, THEN MANTISSA = Ø A ZERO EXPONENT = Ø FLOATING POINT ZERO OUT THE EXPONENT ZERO OUT THE MANTISSA SIGN  SHIFT FPAØ MANTISSA TO RIGHT CLEAR CARRY FLAG  ADD ONE TO EXPONENT ACCUMULATOR SHIFT LS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT NS BYTE SHIFT MS BYTE BRANCH IF NOT YET NORMALIZED GET CURRENT EXPONENT SAVE EXPONENT MODIFIER CAUSED BY NORMALIZATION SUBTRACT ACCUMULATED EXPONENT SAVE AS NEW EXPONENT SET FPAØ = Ø IF THE NORMALIZATION CAUSED

		05 00				PR. 100 T. F. 11 11 T. CO. 1 CHEST OF THE TOTAL OF THE TO
3828	BA5C	25 Ø8	BA5C	BCS	LBA66	BRANCH IF MANTISSA OVERFLOW
3829 383Ø	BA5E	Ø8 63		ASL	FPSBYT	SUB BYTE BIT 7 TO CARRY - USE AS ROUND-OFF FLAG (TRUNCATE THE REST OF SUB BYTE)
3831	BA6Ø	86 ØØ		LDA	#0	CLRA, BUT DO NOT CHANGE CARRY FLAG
3832	BA62	97 63		STA	FPSBYT	CLEAR THE SUB BYTE
3833	BA64	20 0C		BRA	LBA72	GO ROUND-OFF RESULT
3834	BA66	ØC 4F	LBA66	INC	FPØEXP	INCREMENT EXPONENT - MULTIPLY BY 2
3835	BA68	27 28		BEQ	LBA92	OVERFLOW ERROR IF CARRY PAST \$FF
3836	BA6A	06 50		ROR	FPAØ	* SHIFT MANTISSA
3837	BA6C	Ø6 51		ROR	FPAØ+1	* ONE TO
3838	BA6E	Ø6 52		ROR	FPAØ+2	* THE RIGHT -
3839	BA7Ø	06 53		ROR	FPAØ+3	* DIVIDE BY TWO
3840	BA72	24 Ø4	LBA72	BCC	LBA78	BRANCH IF NO ROUND-OFF NEEDED
3841 3842	BA74 BA76	8D ØD 27 EE		BSR BEQ	LBA83 LBA66	ADD ONE TO MANTISSA - ROUND OFF BRANCH IF OVERFLOW - MANTISSA = Ø
3843	BA78	39	LBA78	RTS	LDAUO	BRANCH IF OVERFLOW - MANIISSA - W
3844	DATO	33	* NEGATE		ANTISSA	
3845	BA79	Ø3 54	LBA79	COM	FPØSGN	TOGGLE SIGN OF MANTISSA
3846	BA7B	03 50	LBA7B	COM	FPAØ	* COMPLEMENT ALL 4 MANTISSA BYTES
3847	BA7D	03 51		COM	FPAØ+1	*
3848	BA7F	Ø3 52		COM	FPAØ+2	*
3849	BA81	Ø3 53		COM	FPAØ+3	*
3850					AØ MANTISSA	
3851	BA83	9E 52	LBA83	LDX	FPAØ+2	* GET BOTTOM 2 MANTISSA
3852 3853	BA85 BA87	30 01 9F 52		LEAX STX	1,X FPAØ+2	* BYTES, ADD ONE TO * THEM AND SAVE THEM
3854	BA89	26 Ø6		BNE	LBA91	BRANCH IF NO OVERFLOW
3855	BA8B	9E 5Ø		LDX	FPAØ	* IF OVERFLOW ADD ONE
3856	BA8D	30 01		LEAX	1,X	* TO TOP 2 MANTISSA
3857	BA8F	9F 5Ø		STX	FPAØ	* BYTES AND SAVE THEM
3858	BA91	39	LBA91	RTS		
3859	BA92	C6 ØA	LBA92	LDB	#2*5	'OV' OVERFLOW ERROR
3860	BA94	7E AC 46		JMP	LAC46	PROCESS AN ERROR
3861	BA97	8E ØØ 12	LBA97	LDX	#FPA2-1	POINT X TO FPA2
3862					NTED TO BY (X) TO	
3863					) TIMES. EXIT WITH	
3864 3865					NG DATA SHIFTED OUT (SUB BYTE) AND THE DATA	
3866					(SUB BITE) AND THE DATA OM THE LEFT WILL COME FROM	EDCADY
3867	BA9A	A6 Ø4	LBA9A	LDA	4,X	GET LS BYTE OF MANTISSA (X)
3868	BA9C	97 63		STA	FPSBYT	SAVE IN FPA SUB BYTE
3869	BA9E	A6 Ø3		LDA	3,X	* SHIFT THE NEXT THREE BYTES OF THE
387Ø	BAAØ	A7 Ø4		STA	4,X	* MANTISSA RIGHT ONE COMPLETE BYTE.
3871	BAA2	A6 Ø2		LDA	2,X	*
3872	BAA4	A7 Ø3		STA	3,X	*
3873	BAA6	A6 Ø1		LDA	1,X	*
3874	BAA8	A7 Ø2		STA	2,X	*
3875	BAAA	96 5B		LDA	FPCARY	GET THE CARRY IN BYTE
3876 3877	BAAC BAAE	A7 Ø1 CB Ø8	LBAAE	STA ADDB	1,X #8	STORE AS THE MS MANTISSA BYTE OF (X) ADD 8 TO DIFFERENCE OF EXPONENTS
3878	BABØ	2F E8	LDAAL	BLE	LBA9A	BRANCH IF EXPONENT DIFFERENCE < -8
3879	BAB2	96 63		LDA	FPSBYT	GET FPA SUB BYTE
3880	BAB4	CØ Ø8		SUBB	#8	CAST OUT THE 8 ADDED IN ABOVE
3881	BAB6	27 ØC		BEQ	LBAC4	BRANCH IF EXPONENT DIFFERENCE = Ø
3882			* SHIFT M	ANTISS	A POINTED TO BY (X) TO	
3883			\$ THE RIG	HT (B)	TIMES. OVERFLOW RETAINED	IN ACCA.
3884	BAB8	67 Ø1	LBAB8	ASR	1,X	* SHIFT MANTISSA AND SUB BYTE ONE BIT TO THE RIGHT
3885	BABA	66 Ø2	LBABA	ROR	2,X	*
3886	BABC	66 Ø3		ROR	3,X	*
3887	BABE	66 Ø4		ROR	4,X	*
3888	BACØ			RORA		* ADD ONE TO EVDONENT DIFFEDENCE
3889 389Ø	BAC1	26 F4		INCB	LBAB8	ADD ONE TO EXPONENT DIFFERENCE  BRANCH IF EXPONENTS NOT =
3891			LBAC4	RTS	EBABO	BRANCH IT EXTONENTS NOT -
3892		81 00 00 00 00	LBAC5		\$81,\$00,\$00,\$00,\$00	FLOATING POINT CONSTANT 1.0
3893					,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	
3894			* ARITHME	TIC OP	ERATION (*) JUMPS HERE - M	ULTIPLY
3895			* FPAØ BY	(X) -	RETURN PRODUCT IN FPAØ	
3896		8D 63	LBACA	BSR	LBB2F	MOVE PACKED FPA FROM (X) TO FPA1
3897		27 60			LBB2E	BRANCH IF EXPONENT OF FPAØ = Ø
3898	BACE	8D 78			LBB48	CALCULATE EXPONENT OF PRODUCT
3899					MANTISSA BY FPA1. NORMALI	
3900 3901					TES OF PRODUCT IN FPAØ. TH R BYTES OF THE PRODUCT WIL	
3902			* BE STOR			L
	RADØ	86 ØØ	LBADØ	LDA	#Ø	* ZERO OUT MANTISSA OF FPA2
		97 13	COUDS	STA	FPA2	* ZERO OUT MANTISSA OF FFAZ
		97 14		STA	FPA2+1	*
		97 15		STA		*
3907	BAD8	97 16		STA		*
	BADA	D6 53		LDB	FPAØ+3	GET LS BYTE OF FPAØ
				BSR		MULTIPLY BY FPA1
	BADC					
3910	BADC BADE	D6 63		LDB	FPSBYT	* TEMPORARILY SAVE SUB BYTE 4
391Ø 3911	BADC BADE BAEØ	D6 63 D7 AE		STB	VAE	*
391Ø 3911 3912	BADC BADE BAEØ BAE2	D6 63 D7 AE D6 52		STB LDB	VAE FPAØ+2	* GET NUMBER 3 MANTISSA BYTE OF FPAØ
3910 3911 3912 3913	BADC BADE BAEØ BAE2 BAE4	D6 63 D7 AE D6 52 8D 1A		STB LDB BSR	VAE FPAØ+2 LBBØØ	* GET NUMBER 3 MANTISSA BYTE OF FPAØ MULTIPLY BY FPA1
3910 3911 3912 3913 3914	BADC BADE BAEØ BAE2 BAE4 BAE6	D6 63 D7 AE D6 52 8D 1A D6 63		STB LDB BSR LDB	VAE FPAØ+2 LBBØØ FPSBYT	* GET NUMBER 3 MANTISSA BYTE OF FPAØ
3910 3911 3912 3913 3914	BADC BADE BAEØ BAE2 BAE4 BAE6 BAE8	D6 63 D7 AE D6 52 8D 1A		STB LDB BSR	VAE FPAØ+2 LBBØØ FPSBYT VAD	* GET NUMBER 3 MANTISSA BYTE OF FPAØ MULTIPLY BY FPA1

3917	BAEC	8D 1	,			BSR	LBBØØ		MULTIPLY BY FPA1
3918	BAEE					LDB	FPSBYT		* TEMPORARILY SAVE SUB BYTE 2
	BAFØ					STB	VAC		*
	BAF2					LDB	FPAØ		GET MS BYTE OF FPAØ MANTISSA
3921	BAF4					BSR	LBBØ2		MULTIPLY BY FPA1
	BAF6					LDB	FPSBYT		* TEMPORARILY SAVE SUB BYTE 1
	BAF8					STB	VAB		*
	BAFA					JSR	LBCØB		COPY MANTISSA FROM FPA2 TO FPA0
3925	BAFD					JMP	LBA1C		NORMALIZE FPAØ
3926	BBØØ				LBBØØ	BEQ	LBA97		SHIFT FPA2 ONE BYTE TO RIGHT
3927	BBØ2				LBBØ2	COMA			SET CARRY FLAG
3928							MANTISSA BY ACCB AND		
3929							) FPA2 MANTISSA		
3930	BBØ3	96 1	3		LBBØ3	LDA	FPA2		GET FPA2 MS BYTE
3931	BBØ5					RORB			ROTATE CARRY FLAG INTO SHIFT COUNTER;
3932					*				DATA BIT INTO CARRY
3933	BBØ6	27 2	5			BEQ	LBB2E		BRANCH WHEN 8 SHIFTS DONE
3934	BBØ8	24 1	5			BCC	LBB2Ø		DO NOT ADD FPA1 IF DATA BIT = Ø
3935	BBØA	96 1	5			LDA	FPA2+3		* ADD MANTISSA LS BYTE
3936	BBØC	9B 6	ð			ADDA	FPA1+3		*
3937	BBØE	97 1	5			STA	FPA2+3		*
3938	BB10	96 1	5			LDA	FPA2+2		= ADD MANTISSA NUMBER 3 BYTE
3939	BB12	99 5	F			ADCA	FPA1+2		=
3940	BB14	97 1	5			STA	FPA2+2		=
3941	BB16					LDA	FPA2+1		* ADD MANTISSA NUMBER 2 BYTE
	BB18						FPA1+1		*
	BB1A					STA	FPA2+1		*
	BB1C					LDA	FPA2		= ADD MANTISSA MS BYTE
	BB1E		)			ADCA	FPA1		=
3946	BB2Ø		_		LBB2Ø	RORA			* ROTATE CARRY INTO MS BYTE
	BB21					STA	FPA2		*
	BB23					ROR	FPA2+1		= ROTATE FPA2 ONE BIT TO THE RIGHT
	BB25					ROR	FPA2+2		=
	BB27					ROR	FPA2+3		=
3951	BB29		3			ROR	FPSBYT		= CLEAR CARRY FLAC
3952	BB2B					CLRA	10000		CLEAR CARRY FLAG
3953	BB2C		0		LBB2E	BRA RTS	LBBØ3		KEEP LOOPING
3954	BB2E	39					IMBER FROM (X) TO FPA1		
3955 3956	BB2F	EC @			LBB2F	LDD	1,X		GET TWO MSB BYTES OF MANTISSA FROM
3957	0021	LC 10	L		LDDZI	LDD	1,^		FPA POINTED TO BY X
3958	BB31	07 6				STA	FP1SGN		SAVE PACKED MANTISSA SIGN BYTE
3959	BB33					ORA	#\$8Ø		FORCE BIT 7 OF MSB MANTISSA = 1
3960	BB35					STD	FPA1		SAVE 2 MSB BYTES IN FPA1
3961	BB37	D6 6				LDB	FP1SGN		* GET PACKED MANTISSA SIGN BYTE. EOR W/FPAØ
3962	BB39						FPØSGN		* SIGN - NEW SIGN POSITION IF BOTH OLD SIGNS ALIKE,
3963	BB3B					STB	RESSGN		* NEG IF BOTH OLD SIGNS DIFF. SAVE ADJUSTED
3964	0000	<i>D</i> , 0	-		*	316	KESSUN		* MANTISSA SIGN BYTE
3965	BB3D	EC Ø	3			LDD	3,X		= GET 2 LSB BYTES OF MANTISSA
3966	BB3F					STD	FPA1+2		= AND PUT IN FPA1
3967	BB41					LDA	, χ		* GET EXPONENT FROM (X) AND
3968	BB43					STA	FP1EXP		* PUT IN EXPONENT OF FPA1
3969						LDB	FPØEXP		GET EXPONENT OF FPAØ
	BB45	UO 4							GET EXPUNENT OF FPAU
397Ø	BB45 BB47					RTS	TIBENI		GET EXPONENT OF FPAD
397Ø 3971					* CALCULA	RTS	NENT FOR PRODUCT OF FP	AØ &	
						RTS TE EXPO		AØ &	
3971	BB47 BB48	39 4D				RTS TE EXPO	NENT FOR PRODUCT OF FP	AØ &	
3971 3972	BB47	39 4D			* ENTER W	RTS TE EXPO ITH EXF	NENT FOR PRODUCT OF FP	AØ &	FPA1
3971 3972 3973	BB48 BB49 BB4B	39 4D	5		* ENTER W	RTS TE EXPO ITH EXF TSTA	ONENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA	AØ &	FPA1  TEST EXPONENT OF FPA1  PURGE RETURN ADDRESS & SET FPAØ = Ø  ADD FPA1 EXPONENT TO FPAØ EXPONENT
3971 3972 3973 3974 3975 3976	BB48 BB49 BB4B BB4D	39 4D 27 1 9B 4 46	5		* ENTER W	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA	DNENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA LBB61	AØ &	FPA1  TEST EXPONENT OF FPA1  PURGE RETURN ADDRESS & SET FPAØ = Ø  ADD FPA1 EXPONENT TO FPAØ EXPONENT  ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY
3971 3972 3973 3974 3975 3976 3977	BB48 BB49 BB4B BB4D BB4E	39 4D 27 1 9B 4 46 49	5 F		* ENTER W	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA ROLA	NENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA LBB61 FPØEXP	AØ &	FPA1  TEST EXPONENT OF FPA1  PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAØ
3971 3972 3973 3974 3975 3976 3977 3978	BB48 BB49 BB4B BB4D BB4E BB4F	4D 27 1 9B 4 46 49 28 1	5 F		* ENTER W	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA ROLA BVC	NENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA LBB61 FPØEXP	AØ &	FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL
3971 3972 3973 3974 3975 3976 3977 3978 3979	BB48 BB49 BB4B BB4D BB4E BB4F BB51	4D 27 1 9B 4 46 49 28 1 8B 8	5 F Ø Ø		* ENTER W	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA	DINENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA LBB61 FPØEXP LBB61 #\$80	AØ &	FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT
3971 3972 3973 3974 3975 3976 3977 3978 3979 3980	BB48 BB49 BB4B BB4D BB4E BB4F BB51 BB53	4D 27 1 9B 4 46 49 28 1 8B 8 97 4	5 F Ø Ø		* ENTER W	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA	NENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA LBB61 FPØEXP LBB61 #\$80 FPØEXP	AØ &	FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPA0 = 0 ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT 0 INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$800 BIAS TO EXPONENT SAVE NEW EXPONENT
3971 3972 3973 3974 3975 3976 3977 3978 3979 3980 3981	BB48 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø	5 F Ø Ø F		* ENTER W	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ	NENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP  LBB63	AØ &	FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3979 3980 3981 3982	BB48 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB55	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø 96 6	5 F 3 F 5 2		* ENTER W	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA	DIENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA LBB61 FPØEXP LBB61 #\$80 FPØEXP LBB63 RESSGN	AØ &	FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN
3971 3972 3973 3974 3975 3976 3977 3978 3979 3980 3981 3982 3983	BB48 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø 96 6 97 5	5 F 3 F 5 2		* ENTER W	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA	NENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP  LBB63	AØ &	FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3979 3980 3981 3982 3983 3984	BB48 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB55	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø 96 6 97 5	5 F 3 F 5 2		* ENTER W LBB48	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS	DNENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN		FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3979 3980 3981 3982 3983 3984 3985	BB48 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø 96 6 97 5	5 F 3 F 5 2		* ENTER W LBB48	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS = POSI	DIENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I		FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3979 3980 3981 3982 3983 3984 3985 3986	BB44 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59 BB5B	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø 96 6 97 5 39	5 F Ø F C 2 2		* ENTER W LBB48  * IF FPAØ * = IS NE	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS = POSI GATIVE	INENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPA0 = Ø		FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3978 3980 3981 3982 3983 3984 3985 3986 3986	BB447  BB48 BB49 BB4B BB41 BB4F BB51 BB57 BB59 BB5B	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø 96 6 6 97 5 39	5 F Ø F C 2 2		* ENTER W LBB48	RTS TE EXPO ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS = POSI GATIVE LDA	DIENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I		FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ  GET MANTISSA SIGN OF FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3989 3981 3982 3983 3984 3985 3986 3987 3988	BB448 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59 BB5B	4D 27 1 9B 4 46 49 28 1 8B 8 97 4 27 Ø 96 6 97 5 39 96 5 43	5 5 7 8 8 8 8 8 8 9 1 1 1 1		* ENTER W LBB48  * IF FPAØ * = IS NE	RTS TE EXPO ITH EXP TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS = POSI GATIVE LDA COMA	DNENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB63 RESSGN FPØSSGN  TIVE THEN 'OV' ERROR I THEN FPAØ = Ø FPØSGN		FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BLT 7; BLT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BLAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3980 3981 3982 3983 3984 3985 3986 3986 3988 3988	BB44 BB48 BB49 BB4B BB4F BB51 BB53 BB55 BB57 BB59 BB5B	4D 27 1 98 4 49 28 1 88 8 97 4 27 Ø 96 6 97 5 39 96 5 43 2Ø Ø	66 F F C 22 4		* ENTER W LBB48 * IF FPAØ * = IS NE LBB5C	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS = POSI GATIVE LDA COMA BRA	INENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPAØ = Ø FPØSGN  LBB63		FPA1  TEST EXPONENT OF FPA1  PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BLT 7; BLT Ø INTO CARRY SET OVERFLOW FLAG  BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BLAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ  GET MANTISSA SIGN OF FPAØ  CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ MANTISSA
3971 3972 3973 3974 3975 3976 3977 3978 3979 3981 3982 3983 3984 3985 3986 3987 3988 3989	BB44 BB48 BB49 BB4B BB4F BB51 BB53 BB55 BB57 BB59 BB5B	39 40 27 1 98 4 46 49 28 1 88 8 8 97 4 27 Ø 96 6 97 5 39 96 5 43 2Ø Ø 32 6	5 F F 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS = POSI GATIVE LDA COMA BRA LEAS	DIENT FOR PRODUCT OF FPONENT OF FPA1 IN ACCA  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPA8 = Ø FPØSGN  LBB63 2,5		FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPA0 = 0 ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT 0 INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD 580 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPA0 GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPA0 A0  GET MANTISSA SIGN OF FPA0 CHANGE SIGN OF FPA0 MANTISSA PURGE RETURN ADDRESS FROM STACK
3971 3972 3973 3974 3975 3976 3977 3978 3989 3981 3982 3983 3984 3985 3986 3987 3988 3989 3989 3989	BB44 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59 BB5B	4D 27 1 4 46 49 28 1 88 8 97 4 6 97 5 39 96 5 43 0 32 6 10 2	5 F F C 2 1 1 4 2 2 A FE C	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA BEQ LDA STA RTS GATIVE LDA COMA BRA COMA BRA LEAS LBPL	DNENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 #\$80 #\$80 #\$P0EXP  LBB63 RESSGN FP0SGN  CITUE THEN 'OV' ERROR I THEN FPA0 = Ø FP0SGN  LBB63 LBB63 2,5 LBB63		FPA1  TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN GET MANTISSA SIGN OF FPAØ  CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT
3971 3972 3973 3974 3975 3976 3977 3978 3979 3981 3982 3983 3984 3985 3986 3986 3988 3989 3999 3991 3992	BB44 BB48 BB49 BB4B BB4F BB51 BB53 BB55 BB57 BB59 BB5B	4D 27 1 4 46 49 28 1 88 8 97 4 6 97 5 39 96 5 43 0 32 6 10 2	5 F F C 2 1 1 4 2 2 A FE C	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS POSI GATIVE LDA COMA BRA LEAS LBPL JMP	DNENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPAØ = Ø FPØSGN  LBB63 2,5 LBA99 LBA92	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ  CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR
3971 3972 3973 3974 3975 3976 3977 3978 3989 3981 3982 3983 3984 3985 3986 3987 3988 3989 3989 3989	BB44 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59 BB5B	4D 4D 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	55 FF FF CC CC 22 44 44 44 44 44 44 44 44 44 44 44 44	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67	RTS TE EXPC ITH EXF TSTA BEQ ADDA RORA ROLA BVC ADDA STA BEQ LDA STA RTS POSI GATIVE LDA COMA BRA LEAS LBPL JMP	DNENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 #\$80 #\$80 #\$P0EXP  LBB63 RESSGN FP0SGN  CITUE THEN 'OV' ERROR I THEN FPA0 = Ø FP0SGN  LBB63 LBB63 2,5 LBB63	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ  CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR
3971 3972 3973 3974 3975 3976 3977 3978 3980 3981 3982 3983 3984 3985 3986 3987 3988 3988 3989 3990 3991	BB44 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59 BB5B	39 40 27 1 98 4 46 49 28 1 88 8 8 97 4 27 Ø 6 6 5 39 96 5 43 2Ø Ø 32 6 1 Ø 2 7 E B	5 5 5 F F F F F F F F F F F F F F F F F	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPC ITH EXP TSTA BEQ ADDA RORA BVC ADDA STA BEQ LDA STA EPOSI GATIVE LDA COMA BRA LEAS LBPL LJMP LTIPLY	DIENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPA0 = 0 FPØSGN  LBB63 2,S LBA39 LBA92 BY 10 AND LEAVE RESULT	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ AØ  GET MANTISSA SIGN OF FPAØ CHANGE SIGN OF FPAØ MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPAØ
3971 3972 3973 3974 3975 3976 3977 3978 3980 3981 3982 3983 3984 3985 3986 3987 3988 3989 3999 3991 3992 3993	BB44 BB49 BB4B BB4D BB4E BB51 BB53 BB55 BB57 BB59 BB5B BB5E BB5E BB5E BB61 BB63 BB67 BB6A	39 40 27 1 98 4 46 49 28 18 80 8 43 26 6 10 2 7 E B	5 5 5 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPC ITH EXF TSTA BEQ ADDA ROLA BVC ASTA BEQ LDA STA RTS = POSI GATIVE LDA COMA BRA COMA BRA LEAS LBPL JMP LTIPLY JSR	DNENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  CITUE THEN 'OV' ERROR I THEN FPA0 = Ø FPØSGN  LBB63 2,5 LBA39 LBA39 LBA32 BY 10 AND LEAVE RESULT LBC5F LBB7C	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPA0 = 0 ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPA0 GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPA0 AO  GET MANTISSA SIGN OF FPA0 CHANGE SIGN OF FPA0 MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPA0 MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPA0 TRANSFER FPA0 TO FPA1
3971 3972 3973 3974 3975 3976 3977 3978 3989 3981 3982 3983 3984 3985 3986 3987 3988 3989 3991 3992 3993 3991 3992	BB44 BB49 BB4B BB4D BB4E BB4F BB51 BB53 BB55 BB57 BB59 BB5B BB5E BB5E BB5E BB61 BB63 BB67 BB63 BB67	39 40 27 1 98 4 46 49 28 1 88 8 97 4 27 0 9 96 5 43 20 0 32 6 2 7E B	5 5 5 F F F F F F F F F F F F F F F F F	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPECTION TESTA BEQ ADDA ROLA BVC ADDA STA BEQ LDA STA RTS = POSI GATIVE LDA COMA BRA LEAS LEAS LEBPL JMP LTIPLY JSR BEQ	DNENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  CITUE THEN 'OV' ERROR I THEN FPA0 = Ø FPØSGN  LBB63 2,5 LBA39 LBA39 LBA32 BY 10 AND LEAVE RESULT LBC5F LBB7C	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ  CHANGE SIGN OF FPAØ MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPAØ TRANSFER FPAØ TO FPA1 BRANCH IF EXPONENT = Ø
3971 3972 3973 3974 3975 3976 3977 3988 3981 3982 3983 3985 3986 3987 3988 3989 3990 3991 3992 3993 3994 3995	BB44 BB49 BB44 BB44 BB45 BB53 BB55 BB57 BB59 BB5B BB56 BB56 BB56 BB67 BB63 BB67	40 17 1 98 4 46 49 28 1 88 8 8 97 49 6 6 6 97 5 39 8 8 8 10 27 6 8 8 8 8 27 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPET TSTA BEQ ADDA ROLA BVC ADDA STA BEQ LDA STA RTS = POSIS GATIVE LDA	DIENT FOR PRODUCT OF FPONENT OF FPA1 IN ACCA  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPA0 = 0 FPØSGN  LBB63 2,S LBA39 LBA39 LBA39 LBA92 BY 10 AND LEAVE RESULT LBC5F LBB7C #2	F FP.	TEST EXPONENT OF FPAI PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPAI EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ AØ  GET MANTISSA SIGN OF FPAØ CHANGE SIGN OF FPAØ MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPAØ TRANSFER FPAØ TO FPAI BRANCH IF EXPONENT = Ø ADD 2 TO EXPONENT (TIMES 4)
3971 3972 3973 3974 3975 3976 3977 3978 3988 3981 3982 3983 3984 3985 3986 3987 3988 3999 3991 3992 3993 3994 3994 3995	BB44 BB49 BB48 BB41 BB45 BB53 BB55 BB57 BB59 BB58 BB56 BB56 BB56 BB67 BB63 BB67 BB6A BB67 BB6A BB67	40 27 1 98 4 46 49 28 1 88 8 8 97 4 43 96 6 5 43 26 6 5 10 2 7 8 8 8 8 8 8 27 8 8 8 8 8 8 8 8 8 8	55 57 77 78 78 78 78 78 78 78 78 78 78 78 78	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPE TE EXPE TE STA BEQ ADDA ROLA BVC ADDA STA BEQ LDA STA BEQ LDA STA BEQ LDA	DIENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPA0 = Ø FPØSGN  LBB63 2,S LBA39 LBA92 BB492 BB7 W AND LEAVE RESULT LBC5F LBB7C #2 LBB67	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPA0 = 0 ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT 0 INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD 580 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPA0 GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPA0 AO GET MANTISSA SIGN OF FPA0 CHANGE SIGN OF FPA0 MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPA0 MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPA0 TRANSFER FPA0 TO FPA1 BRANCH IF EXPONENT = 0 ADD 2 TO EXPONENT (TIMES 4) 'OV' ERROR IF EXPONENT > \$FF
3971 3972 3973 3974 3975 3976 3977 3978 3989 3981 3982 3983 3984 3985 3986 3987 3988 3989 3991 3992 3993 3991 3992 3993 3994 3995 3996	BB44 BB49 BB48 BB49 BB4E BB4F BB51 BB53 BB55 BB57 BB59 BB5B BB56 BB56 BB56 BB63 BB67 BB64 BB67 BB67 BB73	39 40 27 1 46 49 28 1 88 8 97 5 39 20 6 6 43 20 6 8 8 8 27 6 8 8 8 27 6 8 8 8 8 9 6 5 6 8 8 8 8 9 6 8 8 8 9 6 8 8 8 9 6 8 8 8 9 6 8 8 8 8	5 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPET TE STA BEQ ADDA ADDA STA BUC ADDA STA BUC LDA STA RTS E POSI LDA COMA BRA LBA LBA LBA LBA LBA LBA LBA LBA LBA LB	DIENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP  LBB63 RESSGN FPØSGN  CITUE THEN 'OV' ERROR I THEN FPA0 = Ø FPØSGN  LBB63 2,5 LBA39 LBA92 BB 10 AND LEAVE RESULT LBCSF LBB7C #2 LBB67 RESSGN	F FP:	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$8Ø BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN GET MANTISSA SIGN OF FPAØ CHANGE SIGN OF FPAØ CHANGE SIGN OF FPAØ MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPAØ TRANSFER FPAØ TO FPA1 BRANCH IF EXPONENT = Ø ADD 2 TO EXPONENT   FF CLEAR RESULT SIGN BYTE
3971 3972 3973 3974 3975 3976 3977 3978 3979 3981 3982 3983 3984 3985 3986 3986 3989 3991 3992 3993 3991 3992 3993 3995 3996 3997	BB44 BB49 BB48 BB49 BB4E BB4F BB51 BB53 BB55 BB55 BB56 BB56 BB56 BB56 BB61 BB63 BB67 BB63 BB67 BB67 BB67 BB67 BB67	39 4D 1 1 9B 4 46 49 28 1 8B 8 97 4 42 7 8 8 1 8 8 97 4 8 1 8 1 97 8 8 8 1 97 8 8 8 97 8 96 6 97 8 8 97 8 98 97 8 98 97 8 98 97 98 98 97 98 98 97 98 98 97 98 98 97 98 98 97 98 98 97 98 98 98 98 98 98 98 98 98 98 98 98 98	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPETITH EXPORT EXPETITH EXPETIT	DIRECT FOR PRODUCT OF FPONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPA0 = Ø FPØSGN  LBB63 2,S LBA39 LBA39 LBA39 LB ADD LEAVE RESULT LBC5F LBB7C #2 LBB67 RESSGN LBB9CD	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPA0 = 0 ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT 0 INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPA0 GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPA0 A0 GET MANTISSA SIGN OF FPA0 CHANGE SIGN OF FPA0 MANTISSA PURGE RETURN ADDRESS FROM STACK ZERO FPA0 MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPA0 TRANSFER FPA0 TO FPA1 BRANCH IF EXPONENT = 0 ADD 2 TO EXPONENT (TIMES 4) 'OV' ERROR IF EXPONENT > \$FF CLEAR RESULT SIGN BYTE ADD FPA1 TO FPA0 (TIMES 5)
3971 3972 3973 3974 3975 3976 3977 3978 3988 3981 3982 3983 3984 3985 3986 3987 3988 3999 3991 3991 3992 3993 3994 3995	BB44 BB49 BB44 BB44 BB45 BB53 BB55 BB57 BB59 BB56 BB67 BB67 BB67 BB67 BB67 BB67 BB67	39 4D 1 1 9B 4 46 49 28 1 8B 8 97 4 42 7 8 8 1 8 8 97 4 8 1 8 1 97 8 8 8 1 97 8 8 8 97 8 96 6 97 8 8 97 8 98 97 8 98 97 8 98 97 98 98 97 98 98 97 98 98 97 98 98 97 98 98 97 98 98 97 98 98 98 98 98 98 98 98 98 98 98 98 98	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	2	* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU	RTS TE EXPET TSTA BEQ ADDA ROLA BVC ADDA STA BEQ LDA STA BEQ LDA STA BEQ LDA STA BEQ LDA	DENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP LBB63 RESSGN FPØSGN  THEN FPA0 = 0 FPØSGN  LBB63 2,S LBA39 LBB63 2,S LBA39 LBB67 RESSGN LBB67 RESSGN FPØSGN	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPA0 = 0 ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT 0 INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD S00 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPA0 GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPA0 A0 GET MANTISSA SIGN OF FPA0 CHANGE SIGN OF FPA0 MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPA0 MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPA0 TRANSFER FPA0 TO FPA1 BRANCH IF EXPONENT = 0 ADD 2 TO EXPONENT (TIMES 4) 'OV' ERROR IF EXPONENT > \$FF CLEAR RESULT SIGN BYTE ADD FPA1 TO FPA0 (TIMES 5) ADD ONE TO EXPONENT (TIMES 5) ADD ONE TO EXPONENT (TIMES 10)
3971 3972 3973 3974 3975 3977 3978 3977 3989 3981 3982 3983 3984 3985 3986 3987 3988 3989 3991 3992 3993 3991 3992 3993 3994 3995 3996 3997 3998 3998	BB449 BB449 BB449 BB445 BB451 BB53 BB557 BB59 BB56 BB561 BB61 BB63 BB67 BB61 BB67 BB67 BB67 BB67 BB73 BB75 BB73 BB75 BB75	39 40 27 1 94 49 46 49 28 1 8 97 4 27 0 39 6 5 42 0 6 27 5 8 8 8 0 6 25 F 6 8 0 8 25 F 6 8 0 8 27 E 8 8 0 6 27 E 8 8 3 3 3 3 3 5 6 8 5 6 6 8 5 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 8 5 7 6 7 6 7 6 7 6 7 6 7 6 7 6 7 6 7 6 7	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5		* IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU LBB6A	RTS TE EXPET	DENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP LBB63 RESSGN FPØSGN  THEN FPA0 = 0 FPØSGN  LBB63 2,S LBA39 LBB63 2,S LBA39 LBB67 RESSGN LBB67 RESSGN FPØSGN	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPA0 = 0 ADD FPA1 EXPONENT TO FPA0 EXPONENT ROTATE CARRY INTO BIT 7; BIT 0 INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD S00 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPA0 GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPA0 A0 GET MANTISSA SIGN OF FPA0 CHANGE SIGN OF FPA0 MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPA0 MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPA0 TRANSFER FPA0 TO FPA1 BRANCH IF EXPONENT = 0 ADD 2 TO EXPONENT (TIMES 4) 'OV' ERROR IF EXPONENT > \$FF CLEAR RESULT SIGN BYTE ADD FPA1 TO FPA0 (TIMES 5) ADD ONE TO EXPONENT (TIMES 5) ADD ONE TO EXPONENT (TIMES 10)
3971 3972 3973 3974 3975 3976 3977 3978 3980 3981 3982 3983 3984 3985 3986 3987 3988 3999 3991 3999 3991 3992 3994 3995 3997 3998 3997 3998 4000 4001 4002 4002 4002 4004	BB447  BB48 BB49 BB4E BB4F BB53 BB55 BB57 BB57 BB61 BB63 BB67 BB6A BB6B BB67 BB6A BB67 BB73 BB73 BB73 BB73 BB73 BB77 BB70 BB70	39 4D 1 1 9B 4 6 49 28 1 8 97 4 27 0 6 6 9 39 20 0 2 7 E B BD B B B B B B B B B B B B B B B B	55 56 57 57 57 57 57 57 57 57 57 57 57 57 57		* IF FPAØ * = IS NE LBB5C LBB61 LBB63 LBB67 * FAST MU LBB6A	RTS TE EXPET TESTA BEQ ADDA ROLA BVC ADDA STA BEQ LDA STA RTS SIA	DNENT FOR PRODUCT OF FP ONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPAØ = Ø FPØSGN  LBB63 2,5 LBA39 LBA39 LBA39 LBA39 LBA39 LBA56 LB65F LB87C #2 LB667 RESSGN RESSGN FPØEXP LB667 \$84,\$20,\$00,\$00,\$00	F FP.	TEST EXPONENT OF FPA1 PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPA1 EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ AØ  GET MANTISSA SIGN OF FPAØ CHANGE SIGN OF FPAØ MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPAØ TRANSFER FPAØ TO FPA1 BRANCH IF EXPONENT = Ø ADD 2 TO EXPONENT (TIMES 4) 'OV' ERROR IF EXPONENT > \$FF CLEAR RESULT SIGN BYTE ADD FPAØ (TIMES 5) ADD ONE TO EXPONENT (TIMES 1Ø) 'OV' ERROR IF EXPONENT > \$FF FLOATING POINT CONSTANT 1Ø
3971 3972 3973 3974 3975 3976 3977 3978 3980 3981 3982 3983 3984 3985 3986 3987 3988 3999 3991 3992 3993 3994 3995 3997 3996 3997 3998 3999 4000 4001 4002 4002	BB449 BB449 BB449 BB445 BB451 BB53 BB557 BB59 BB56 BB561 BB61 BB63 BB67 BB61 BB67 BB67 BB67 BB67 BB73 BB75 BB73 BB75 BB75	39 4D 1 1 9B 4 6 49 28 1 8 97 4 27 0 6 6 9 39 20 0 2 7 E B BD B B B B B B B B B B B B B B B B	55 56 57 57 57 57 57 57 57 57 57 57 57 57 57		* ENTER W LBB48  * IF FPAØ * = IS NE LBB5C  LBB61 LBB63 LBB67 * FAST MU LBB6A	RTS TE EXPET TESTA BEQ ADDA ROLA BVC ADDA STA BEQ LDA STA BEQ LDA STA BEQ LDA STA BEQ LDA	DIENT FOR PRODUCT OF FP PONENT OF FPA1 IN ACCA  LBB61 FPØEXP  LBB61 #\$80 FPØEXP LBB63 RESSGN FPØSGN  TIVE THEN 'OV' ERROR I THEN FPA0 = 0 FPØSGN  LBB63 2,S LBA39 LBB67 LBA92 BY 10 AND LEAVE RESULT LBC5F LB7C #2 LBB67 RESSGN LB9C0 FPØEXP LB967 \$84,\$20,\$00,\$00,\$00	F FP.	TEST EXPONENT OF FPAI PURGE RETURN ADDRESS & SET FPAØ = Ø ADD FPAI EXPONENT TO FPAØ EXPONENT ROTATE CARRY INTO BIT 7; BIT Ø INTO CARRY SET OVERFLOW FLAG BRANCH IF EXPONENT TOO LARGE OR SMALL ADD \$80 BIAS TO EXPONENT SAVE NEW EXPONENT SET FPAØ GET MANTISSA SIGN SAVE AS MANTISSA SIGN OF FPAØ  CHANGE SIGN OF FPAØ MANTISSA  PURGE RETURN ADDRESS FROM STACK ZERO FPAØ MANTISSA SIGN & EXPONENT 'OV' OVERFLOW ERROR FPAØ TRANSFER FPAØ TO FPAI BRANCH IF EXPONENT = Ø ADD 2 TO EXPONENT (TIMES 4) 'OV' ERROR IF EXPONENT > \$FF CLEAR RESULT SIGN WITE ADD FPAI TO FPAØ (TIMES 5) ADD ONE TO EXPONENT (TIMES 10) 'OV' ERROR IF EXPONENT > \$FF

4006	BB85	8E	RR	7 D		LDX	#LBB7D		POINT TO FLOATING POINT CONSTANT 10
4007	BB88	5F	00	, 5		CLRB	#20070		ZERO MANTISSA SIGN BYTE
4008	BB89	D7	62		LBB89	STB	RESSGN		STORE THE QUOTIENT MANTISSA SIGN BYTE
4009	BB8B	BD	ВС	14		JSR	LBC14		UNPACK AN FP NUMBER FROM (X) INTO FPAØ
4010	BB8E	80				FCB	SKP2		SKIP TWO BYTES
4011								NORMALIZED QUOT	
4012	BB8F	8D	9 E		LBB8F	BSR	LBB2F		GET FP NUMBER FROM (X) TO FPA1
4013					+ 40171145		DATION ( ()	TUMBS HERE DAY	INDE FOAT DV FOAG (FNTED HITH
4014 4015								AND FLAGS SET B	IDE FPA1 BY FPAØ (ENTER WITH
4016					LATONEN	1 01 11	AI IN ACCA	AND ILAGS SET D	131A)
4017					* DIVIDE	FPA1 BY	' FPAØ		
4018	BB91	27	73		LBB91	BEQ	LBCØ6		'/0' DIVIDE BY ZERO ERROR
4019	BB93	ØØ	4 F			NEG	FPØEXP		GET EXPONENT OF RECIPROCAL OF DIVISOR
4020	BB95	8D	В1			BSR	LBB48		CALCULATE EXPONENT OF QUOTIENT
4021	BB97	ØC				INC	FPØEXP		INCREMENT EXPONENT
4022	BB99	27				BEQ	LBB67		'OV' OVERFLOW ERROR
4023	BB9B	8E	00	13	*	LDX	#FPA2		POINT X TO MANTISSA OF FPA2 - HOLD
4024 4025	BB9E	C6	αı		*	LDB	#4		TEMPORARY QUOTIENT IN FPA2 5 BYTE DIVIDE
4026	BBAØ	D7				STB	TMPLOC		SAVE BYTE COUNTER
4027		C6				LDB	#1		SHIFT COUNTER-AND TEMPORARY QUOTIENT BYTE
4028					* COMPARE			FPA1 MANTISSA -	
4029							IF FPA1 >		
4030	BBA4	96	50		LBBA4	LDA	FPAØ		* COMPARE THE TWO MS BYTES
4031	BBA6	91	5 D			CMPA	FPA1		* OF FPAØ AND FPA1 AND
4032		26				BNE	LBBBD		* BRANCH IF <>
4033	BBAA					LDA	FPAØ+1		= COMPARE THE NUMBER 2
4034	BBAC	91					FPA1+1		= BYTES AND
4035 4036	BBAE BBBØ	26 96				BNE	LBBBD		= BRANCH IF <> * COMPARE THE NUMBER 3
4030 4037		91				LDA CMPA	FPAØ+2 FPA1+2		* BYTES AND
4038	BBB4	26				BNE	LBBBD		* BRANCH IF <>
4039	BBB6	96				LDA	FPAØ+3		= COMPARE THE LS BYTES
4040		91					FPA1+3		= AND BRANCH
4041	BBBA	26	01			BNE	LBBBD		= IF ⇔
4042	BBBC	43				COMA			SET CARRY FLAG IF FPAØ = FPA1
4043	BBBD	1 F	Α8		LBBBD	TFR	CC,A		SAVE CARRY FLAG STATUS IN ACCA; CARRY
4044					*				CLEAR IF FPAØ > FPA1
4045		59				ROLB			ROTATE CARRY INTO TEMPORARY QUOTIENT BYTE
4046		24				BCC	LBBCC		CARRY WILL BE SET AFTER 8 SHIFTS
4Ø47 4Ø48	BBC2	ØA				STB	, X+		SAVE TEMPORARY QUOTIENT
4048	BBC4 BBC6	2B				DEC BMI	TMPLOC LBBFC		DECREMENT BYTE COUNTER BRANCH IF DONE
4050		27				BEQ	LBBF8		BRANCH IF LAST BYTE
4051		C6				LDB	#1		RESET SHIFT COUNTER AND TEMPORARY QUOTIENT BYTE
4052		1F			LBBCC	TFR	A,CC		RESTORE CARRY FLAG AND
4053		25				BCS	LBBDE		BRANCH IF FPAØ =< FPA1
4054	BBDØ	Ø8	60		LBBDØ	ASL	FPA1+3		* SHIFT FPA1 MANTISSA 1 BIT TO LEFT
4055	BBD2	Ø9	5 F			ROL	FPA1+2		*
4056	BBD4	Ø9				ROL	FPA1+1		*
4057	BBD6	Ø9				ROL	FPA1		*
4058	BBD8	25				BCS	LBBBD		BRANCH IF CARRY - ADD ONE TO PARTIAL QUOTIENT
4059 4060	BBDA	2B	LB			BMI	LBBA4		IF MSB OF HIGH ORDER MANTISSA BYTE IS SET, CHECK THE MAGNITUDES OF FPAØ, FPA1
4061	BBDC	20	DΕ			BRA	LBBBD		CARRY IS CLEAR, CHECK ANOTHER BIT
4062	DDDC	LU	01		* SHRTRAC			- LEAVE RESULT I	
4063	BBDE	96	60		LBBDE	LDA	FPA1+3	22/112 1120021 1	* SUBTRACT THE LS BYTES OF MANTISSA
4064	BBEØ	90	53			SUBA	FPAØ+3		*
4065	BBE2	97	60			STA	FPA1+3		*
4066	BBE4					LDA	FPA1+2		= THEN THE NEXT BYTE
4067	BBE6						FPAØ+2		=
4068	BBE8					STA	FPA1+2		= * AND THE NEVT
4069 4070	BBEA BBEC					LDA SRCA	FPA1+1 FPAØ+1		* AND THE NEXT
4070	BBEE						FPAU+1 FPA1+1		*
4072	BBFØ					LDA	FPA1		= AND FINALLY, THE MS BYTE OF MANTISSA
4073	BBF2					SBCA			=
4074	BBF4					STA	FPA1		=
4075	BBF6	20	D8			BRA	LBBDØ		GO SHIFT FPA1
4076	BBF8	С6	40		LBBF8	LDB	#\$40		USE ONLY TWO BITS OF THE LAST BYTE (FIFTH)
4077	BBFA						LBBCC		GO SHIFT THE LAST BYTE
4078	BBFC				LBBFC	RORB			* SHIFT CARRY (ALWAYS SET HERE) INTO
4079	BBFD					RORB			* BIT 5 AND MOVE
4080 4081	BBFE BBFF					RORB STB	FDCDVT		* BITS 1,0 TO BITS 7,6 SAVE SUB BYTE
4081 4082	BCØ1						FPSBYT LBCØB		MOVE MANTISSA OF FPA2 TO FPAØ
4082	BCØ3						LBC0B LBA1C		NORMALIZE FPAØ
4084	BCØ6			10	LBCØ6	LDB	#2*10		'/Ø' ERROR
4085	BCØ8			46			LAC46		PROCESS THE ERROR
4Ø86					* COPY MA		FROM FPA2	TO FPAØ	
4Ø87	BCØB	9 E	13		LBCØB		FPA2		* MOVE TOP 2 BYTES
4Ø88	BCØD						FPAØ		*
4Ø89	BCØF						FPA2+2		= MOVE BOTTOM 2 BYTES
4090	BC11						FPAØ+2		=
4091	BC13	39			+ 0007 4	RTS	ED NUMBER	EDOM (V) TO FOR	
4Ø92 4Ø93	BC14	3./	Ø2		* COPY A	PACKED		FROM (X) TO FPAØ	SAVE ACCA
4093	BC14				LDUIT		1,X		GET TOP TWO MANTISSA BYTES
	-	-	-				•		

1005	BC18	9.7	5.4		ΛТ2	FPØSGN	SAVE MS BYTE OF MANTISSA AS MANTISSA SIGN
	BC1A				ORA		UNPACK MS BYTE
	BC1C				STD		SAVE UNPACKED TOP 2 MANTISSA BYTES
4098	BC1E	ØF	63		CLR	FPSBYT	CLEAR MANTISSA SUB BYTE
	BC2Ø				LDB		GET EXPONENT TO ACCB
	BC22				LDX		* MOVE LAST 2
	BC24					FPAØ+2	* MANTISSA BYTES
	BC26					FPØEXP	SAVE EXPONENT
4103	BC28	35	82		PULS	A,PC	RESTORE ACCA AND RETURN
	BC2A	8F	aa 45	LBC2A	I DY	#V45	POINT X TO MANTISSA OF FPA4
	BC2D			LDCZA		LBC35	MOVE FPAØ TO FPA4
	BC2F			LBC2F		#V40	POINT X TO MANTISSA OF FPA3
	BC32					SKP2	SKIP TWO BYTES
	BC33		3B	LBC33		VARDES	POINT X TO VARIABLE DESCRIPTOR IN VARDES
4110				* PACK FP	AØ AND	MOVE IT TO ADDRESS IN X	
4111	BC35	96	4 F	LBC35	LDA	FPØEXP	* COPY EXPONENT
	BC37				STA		*
	BC39					FPØSGN	GET MANTISSA SIGN BIT
	BC3B				ORA		MASK THE BOTTOM 7 BITS
	BC3D					FPAØ	AND BIT 7 OF MANTISSA SIGN INTO BIT 7 OF MS BYTE
	BC3F BC41					1,X FPAØ+1	SAVE MS BYTE * MOVE 2ND MANTISSA BYTE
	BC43				STA		*
	BC45					FPAØ+2	= MOVE BOTTOM 2 MANTISSA BYTES
	BC47				STU		=
	BC49				RTS		
4122				* MOVE FP	A1 T0 I	FPAØ RETURN W/MANTISSA SIGN	I IN ACCA
	BC4A					FP1SGN	* COPY MANTISSA SIGN FROM
	BC4C			LBC4C		FPØSGN	* FPA1 TO FPAØ
	BC4E					FP1EXP	= COPY EXPONENT + MS BYTE FROM
	BC5Ø					FPØEXP	= FPA1 TO FPAØ
	BC52 BC54					FPSBYT FPA1+1	CLEAR MANTISSA SUB BYTE * COPY 2ND MANTISSA BYTE
	BC54					FPAØ+1	* FROM FPA1 TO FPAØ
	BC58					FPØSGN	GET MANTISSA SIGN
	BC5A					FPA1+2	* COPY 3RD AND 4TH MANTISSA BYTE
	BC5C					FPAØ+2	* FROM FPA1 TO FPAØ
4133	BC5E	39			RTS		
4134				* TRANSFE			
	BC5F			LBC5F			* TRANSFER EXPONENT & MS BYTE
	BC61					FP1EXP	*
	BC63					FPAØ+1	= TRANSFER MIDDLE TWO BYTES
	BC65					FPA1+1	+ TRANSFER ROTTON THO RVIES
	BC67 BC69					FPAØ+3 FPA1+3	* TRANSFER BOTTOM TWO BYTES
	BC6B		O D		TSTA	FFAITS	SET FLAGS ACCORDING TO EXPONENT
	BC6C				RTS		SET TENUS ACCORDING TO EXTONERT
4143				* CHECK F		ETURN ACCB = Ø IF FPAØ = Ø,	
4144				* ACCB =	\$FF IF	FPAØ = NEGATIVE, ACCB = 1	IF FPAØ = POSITIVE
4145	BC6D	D6	4 F	LBC6D		FPØEXP	GET EXPONENT
	BC6F					LBC79	BRANCH IF FPAØ = Ø
	BC71			LBC71		FPØSGN	GET SIGN OF MANTISSA
	BC73			LBC73	ROLB	##FF	BIT 7 TO CARRY NEGATIVE FLAG
	BC74 BC76					#\$FF LBC79	BRANCH IF NEGATIVE MANTISSA
	BC78				NEGB	10079	ACCB = 1 IF POSITIVE MANTISSA
	BC79			LBC79	RTS		ACCE - I II TOSTITE HANTISA
4153							
4154				* SGN			
4155	BC7A	8D	F1	SGN	BSR	LBC6D	SET ACCB ACCORDING TO SIGN OF FPAØ
4156				* CONVERT		NED NUMBER IN ACCB INTO A F	
4157	BC7C			LBC7C	STB	FPAØ	SAVE ACCB IN FPAØ
4158	BC7E				CLR	FPAØ+1	CLEAR NUMBER 2 MANTISSA BYTE OF FPAØ
	BC8Ø BC82			LBC82		#\$88 FPAØ	EXPONENT REQUIRED IF FPAØ IS TO BE AN INTEGER GET MS BYTE OF MANTISSA
	BC84			LBCOZ		#\$80	SET CARRY IF POSITIVE MANTISSA
	BC84			LBC86		FPØEXP	SAVE EXPONENT
	BC88			25000		ZERO	* ZERO OUT ACCD AND
	BC8A					FPAØ+2	* BOTTOM HALF OF FPAØ
4165	BC8C	97	63		STA	FPSBYT	CLEAR SUB BYTE
4166	BC8E	97	54		STA	FPØSGN	CLEAR SIGN OF FPAØ MANTISSA
4167	BC9Ø	7 E	BA 18		JMP	LBA18	GO NORMALIZE FPAØ
4168							
4169		~-		* ABS			
	BC93		54	ABS		FPØSGN	FORCE MANTISSA SIGN OF FPAØ POSITIVE
4171 4172	BC95	39		* COMPADE	RTS A DACI	VED FLOATING DOINT NUMBER 5	DOINTED TO
4172						KED FLOATING POINT NUMBER F UNPACKED FP NUMBER IN FPAØ.	
						AND ACCB = Ø, IF EQUAL; AC	
4173 4174 4175					(X); A(	CCB = \$FF IF FPAØ < (X)	
4174 4175	BC96	E6	84		(X); A( LDB		CHECK EXPONENT OF (X)
4174 4175 4176	BC96 BC98			* FPAØ >	LDB		CHECK EXPONENT OF (X) BRANCH IF FPA = Ø
4174 4175 4176 4177 4178	BC98 BC9A	27 E6	D3 Ø1	* FPAØ >	LDB BEQ	, Х	
4174 4175 4176 4177 4178 4179	BC98 BC9A BC9C	27 E6 D8	D3 Ø1 54	* FPAØ >	LDB BEQ LDB EORB	,X LBC6D 1,X FPØSGN	BRANCH IF FPA = Ø GET MS BYTE OF MANTISSA OF (X) EOR WITH SIGN OF FPAØ
4174 4175 4176 4177 4178 4179 4180	BC98 BC9A	27 E6 D8	D3 Ø1 54	* FPAØ > LBC96	LDB BEQ LDB EORB BMI	,X LBC6D 1,X FPØSGN LBC71	BRANCH IF FPA = Ø GET MS BYTE OF MANTISSA OF (X) EOR WITH SIGN OF FPAØ BRANCH IF SIGNS NOT =
4174 4175 4176 4177 4178 4179 4180 4181	BC98 BC9A BC9C	27 E6 D8	D3 Ø1 54	* FPAØ > LBC96	LDB BEQ LDB EORB BMI FPAØ	,X LBC6D 1,X FPØSGN LBC71 WITH FP NUMBER POINTED TO B	BRANCH IF FPA = Ø GET MS BYTE OF MANTISSA OF (X) EOR WITH SIGN OF FPAØ BRANCH IF SIGNS NOT =
4174 4175 4176 4177 4178 4179 4180	BC98 BC9A BC9C	27 E6 D8 2B	D3 Ø1 54 D1	* FPAØ > LBC96	LDB BEQ LDB EORB BMI FPAØ V	,X LBC6D 1,X FPØSGN LBC71	BRANCH IF FPA = Ø GET MS BYTE OF MANTISSA OF (X) EOR WITH SIGN OF FPAØ BRANCH IF SIGNS NOT =

4184	BCA2	E1 84		СМРВ	, χ	* FPAØ, COMPARE TO EXPONENT OF
4185	BCA4	26 1D		BNE	LBCC3	* (X) AND BRANCH IF <>.
4186	BCA6	E6 Ø1		LDB	1,X	* GET MS BYTE OF (X), KEEP ONLY
4187	BCA8			ORB	#\$7F	* THE SIGN BIT - 'AND' THE BOTTOM 7
4188		D4 50			FPAØ	* BITS OF FPAØ INTO ACCB
4189 4190		E1 Ø1 26 13		CMPB BNE	1,X LBCC3	= COMPARE THE BOTTOM 7 BITS OF THE MANTISSA = MS BYTE AND BRANCH IF <>
4191		D6 51		LDB	FPAØ+1	* COMPARE 2ND BYTE
4192		E1 Ø2		CMPB		* OF MANTISSA,
4193	BCB4	26 ØD		BNE	LBCC3	* BRANCH IF <>
4194		D6 52		LDB	FPAØ+2	= COMPARE 3RD BYTE
4195 4196		E1 Ø3 26 Ø7		CMPB BNE	3,X LBCC3	= OF MANTISSA,
4197		D6 53		LDB	FPAØ+3	= BRANCH IF <> * SUBTRACT LS BYTE
4198		EØ Ø4		SUBB		* OF (X) FROM LS BYTE OF
4199	BCCØ	26 Ø1		BNE	LBCC3	* FPAØ, BRANCH IF <>
4200	BCC2			RTS		RETURN IF FP (X) = FPAØ
4201 4202	BCC3	56 D8 54	LBCC3	RORB	FPØSGN	SHIFT CARRY TO BIT 7; CARRY SET IF FPAØ < (X) TOGGLE SIZE COMPARISON BIT IF FPAØ IS NEGATIVE
4202		20 AB		BRA	LBC73	GO SET ACCB ACCORDING TO COMPARISON
4204			* DE-NORM			UNTIL THE BINARY POINT IS TO THE RIGHT
4205			* OF THE	LEAST	SIGNIFICANT BYTE OF THE MAN	NTISSA
4206		D6 4F	LBCC8	LDB	FPØEXP	GET EXPONENT OF FPAØ
4207		27 3D		BEQ	LBDØ9	ZERO MANTISSA IF FPAØ = Ø
42Ø8 42Ø9	вссс	CØ AØ	*	SUBB	#\$AØ	SUBTRACT \$AØ FROM FPAØ EXPONENT T THIS WILL YIELD THE NUMBER OF SHIFTS REQUIRED TO DENORMALIZE FPAØ. WHEN
4210			*			THE EXPONENT OF FPAØ IS = ZERO, THEN THE BINARY POINT
4211			*			WILL BE TO THE RIGHT OF THE MANTISSA
4212	BCCE	96 54		LDA	FPØSGN	TEST SIGN OF FPAØ MANTISSA
4213		2A Ø5		BPL	LBCD7	BRANCH IF POSITIVE
4214		Ø3 5B		COM	FPCARY	COMPLEMENT CARRY IN BYTE
4215 4216		BD BA 7B 8E ØØ 4F	LBCD7	JSR LDX	LBA7B #FPØEXP	NEGATE MANTISSA OF FPAØ POINT X TO FPAØ
4217		C1 F8	25057	CMPB		EXPONENT DIFFERENCE < -8?
4218	BCDC	2E Ø6		BGT	LBCE4	YES
4219		BD BA AE		JSR	LBAAE	SHIFT FPAØ RIGHT UNTIL FPAØ EXPONENT = \$AØ
422Ø 4221		ØF 5B		CLR RTS	FPCARY	CLEAR CARRY IN BYTE
4221	BCE3 BCE4		LBCE4	CLR	FPCARY	CLEAR CARRY IN BYTE
4223		96 54	25021	LDA	FPØSGN	* GET SIGN OF FPAØ MANTISSA
4224	BCE8	49		ROLA		* ROTATE IT INTO THE CARRY FLAG
4225	BCE9	06 50		ROR	FPAØ	ROTATE CARRY (MANTISSA SIGN) INTO BIT 7
4226			*			OF LS BYTE OF MANTISSA
4227	RCFR	7E BA BA		JMP	LBABA	
				0111	EDADA	DE-NORMALIZE FPAØ
4228 4229			* INT	0111	LUNUN	DE-NORMALIZE FFAB
4228						AØ - THAT IS IT WILL SHIFT THE BINARY POINT
4228 4229			* THE INT * TO THE	STATE	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE
4228 4229 4230 4231 4232			* THE INT * TO THE * THIS IS	STATE EXTREM DONE	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL (	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT
4228 4229 4230 4231 4232 4233			* THE INT * TO THE * THIS IS * BYTES (	STATE EXTREM DONE	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. AT	AØ – THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION
4228 4229 4230 4231 4232 4233 4234			* THE INT * TO THE * THIS IS * BYTES (	STATE EXTREM DONE	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL (	AØ – THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION
4228 4229 4230 4231 4232 4233	BCEE	D6 4F	* THE INT * TO THE * THIS IS * BYTES (	STATE EXTREM DONE	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. AT	AØ – THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237	BCFØ	D6 4F C1 AØ	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	T STATE EXTREM DONE OF THE HE INTE LDB CMPB	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. AT SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT IT THE CONCLUSION OF THE DE-NORMALIZATION MAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238	BCFØ BCF2	D6 4F C1 AØ 24 1D	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM DONE THE INTE LDB CMPB BCC	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. AT GER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239	BCFØ	D6 4F C1 AØ 24 1D	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	T STATE EXTREM DONE OF THE HE INTE LDB CMPB	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. AT SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ == 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240	BCFØ BCF2 BCF4	D6 4F C1 AØ 24 1D 8D D2	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM DONE OF THE HE INTE LDB CMPB BCC BSR	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. A' GER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239	BCFØ BCF2 BCF4	D6 4F C1 AØ 24 1D 8D D2 D7 63	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM DONE THE INTE LDB CMPB BCC	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. AT GER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ == 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4243	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM DONE F THE HE INTE LDB CMPB BCC BSR STB LDA STB	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ. A' GER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCCB FPSBYT FPØSGN FPØSGN	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4243 4244	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM S DONE OF THE INTE LDB CMPB BCC BSR STB LDA STB SUBA	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8 FPSBYT FPØSGN FPØSGN #\$8Ø	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4244 4244 4244	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM SOONE OF THE INTE LDB CMPB BCC BSR STB LDA STB SUBALDA	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8 FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ	A8 - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ == 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4243 4244 4245 4245	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM SO DONE OF THE HE INTE LDB BCC BSR STB LDA STB SUBA LDA STA	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$8Ø #\$\$AØ FPØEXP	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4244 4244 4244	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM SOONE OF THE INTE LDB CMPB BCC BSR STB LDA STB SUBALDA	MENT WILL "DENORMALIZE" FP/ E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8 FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ	A8 - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ == 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4242 4243 4244 4244 4244 4244	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2 BDØ4	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM DONE OF THE HE INTE  LDB CMPB BCC BSR  STB LDA STB SUBA LDA STA LDA	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8 FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ #\$AØ #\$AØ #\$AØ #\$AØ #\$AØ #\$AØ #\$A	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4249 4241 4244 4243 4244 4244 4244 4245 4247 4246 4247 4248 4249 4250	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2 BDØ4 BDØ6	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18	* THE IN' * TO THE * THIS IS * BYTES ( * ONLY TH * INT	STATE EXTREM DONE OF THE LDB CMPB BCC BSR STB LDA STB LDA STB LDA STA	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL INTEGER PORTION OF FPAØ. AT SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8 FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ FPFØSRP FPØSRP FPØEXP FPØEXP FPAØ+3 CHARAC LBA18	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4240 4241 4242 4243 4244 4245 4244 4245 4247 4248 4249 4249 4249	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØ9 BDØ2 BDØ4 BDØ6	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18	* THE INT * TO THE * THIS IS * BYTES ( * ONLY THE	STATE EXTREM DONE DF THE LDB CMPB BCC BSR STB LDA STB SUBA LDA STA JMP STB	MENT WILL "DENORMALIZE" FP/E E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI  FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ FPØEXP #\$AØ CHARAC LBA18	A8 - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4243 4244 4245 4246 4247 4248 4249 4251 4251 4252	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØ9 BDØ2 BDØ4 BDØ6	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51	* THE IN' * TO THE * THIS IS * BYTES ( * ONLY TH * INT	STATE EXTREM DONE DONE THE LDB CMPB BCC BSR  STB LDA STB LDA STB LDA STA LDA S	MENT WILL "DENORMALIZE" FPJE E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8 FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ FPØEXP FPAØH3 CHARAC LBA18 FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ FPAØ	A8 - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB *
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4248 4249 4241 4242 4243 4244 4245 4246 4247 4248 4249 4250 4250 4251	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2 BDØ4 BDØ6 BDØ9 BDØB BDØD	D6 4F C1 AØ 24 1D 8D D2  D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18  D7 50 D7 51 D7 52	* THE IN' * TO THE * THIS IS * BYTES ( * ONLY TH * INT	STATE EXTREM GOOD EXTREM GOOD EXTREM GOOD EXTREM GOOD EXTREM EXTR	MENT WILL "DENORMALIZE" FPJE E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN #\$8Ø #\$8Ø #\$8Ø FPØEXP FPAØ+3 CHARAC LBA18 FPAØ FPAØ+1 FPAØ FPAØ+2	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240 4241 4242 4243 4244 4245 4246 4247 4248 4249 4251 4251 4252	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2 BDØ4 BDØ6 BDØ9 BDØB BDØD	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53	* THE IN' * TO THE * THIS IS * BYTES ( * ONLY TH * INT	STATE EXTREM DONE DONE THE LDB CMPB BCC BSR  STB LDA STB LDA STB LDA STA LDA S	MENT WILL "DENORMALIZE" FPJE E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN #\$8Ø #\$8Ø #\$8Ø FPØEXP FPAØ+3 CHARAC LBA18 FPAØ FPAØ+1 FPAØ FPAØ+2	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB *
4228 4229 4230 4231 4232 4233 4234 4235 4236 4238 4238 4239 4241 4242 4243 4244 4245 4245 4246 4247 4248 4249 4250 4250 4251 4252 4253 4254 4255 4255 4255 4255 4255	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2 BDØ4 BDØ6 BDØ9 BDØB BDØD BDØF	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53	* THE IN* * TO THE IN* * TO THE IN* * THIS IN* * BYTES ( * ONLY THINT  *  LBD09	STATE EXTREM (S DONE EXTREM (S DONE EXTREM (S DONE EXTREM	MENT WILL "DENORMALIZE" FPJE E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$8Ø #\$ABØ FPØEXP FPAØ+3 CHARAC LBB18  FPAØ FPAØ+1 FPAØ+2 FPAØ+3	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB *
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4240 4241 4242 4244 4245 4246 4247 4248 4249 4250 4251 4250 4251 4252 4255 4255 4255 4255 4255 4255	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2 BDØ4 BDØ6 BDØ9 BDØB BDØD BDØF BDØF	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM S DONE COMPB BCC BSR STB LDA STB SUBA LDA STA JMP STB	MENT WILL "DENORMALIZE" FPA E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8 FPSBYT FPØSGN FPØSGN #\$BØ #\$AØ #\$AØ #\$AØ FPAØ+3 CHARAC LBA18 FPAØ+1 FPAØ+2 FPAØ+3	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * * * * * * * * * * * * * * * *
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4249 4241 4242 4243 4244 4245 4244 4245 4245 4251 4251 4252 4253 4254 4255 4256 4257 4258	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØØ BDØ2 BDØ4 BDØ9 BDØ9 BDØ9 BDØ9 BDØ5 BDØ11	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 50 D7 51 D7 52 D7 53 39	* THE IN* * TO THE IN* * TO THE IN* * THIS IN* * BYTES ( * ONLY THINT  *  LBD09	STATE EXTREM DONE DONE DOT THE ELDB BCC BSR STB LDA STA LDA STA JMP STB	MENT WILL "DENORMALIZE" FP/E E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ #\$AØ FPØEXP FPAØAB CHARAC LBA18  FPAØ FPAØB+1 FPAØ+2 FPAØ+3 STRING TO FLOATING POINT ZERO	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * * * * * * * * * * * * * * * *
4228 4229 4230 4231 4232 4233 4234 4235 4236 4238 4239 4240 4241 4242 4243 4244 4245 4246 4247 4248 4251 4251 4251 4253 4255 4255 4255 4255 4255 4255 4255	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BDØØ BDØ2 BDØ4 BDØ6 BDØ9 BDØB BDØF BDI1	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 80 80 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53 39 9E 8A 9F 54	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM (S DONE ) F THE LOB BCC CMPB BCC BSR LDA STB LDA STA STA STA STA STA STA STA STA STA ST	MENT WILL "DENORMALIZE" FP/E E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI  FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$AØ #\$AØ FPØEXP FPAØ+3 CHARAC LBA18  FPAØ FPAØ+3 FPAØ+3 FPAØ+3 FPAØ+3 FPAØ+3 STRING TO FLOATING POINT ZERO FPØSGN	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * * * * * * * * * * * * * * * *
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4249 4241 4242 4243 4244 4245 4244 4245 4245 4251 4251 4252 4253 4254 4255 4256 4257 4258	BCFØ BCF2 BCF4 BCF6 BCF8 BCFC BCFE BDØØ BDØ2 BDØ4 BDØ6 BDØ9 BDØB BDØD BDØF BDI1	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 50 D7 51 D7 52 D7 53 39	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM DONE DONE DOT THE ELDB BCC BSR STB LDA STA LDA STA JMP STB	MENT WILL "DENORMALIZE" FP/E E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ #\$AØ FPØEXP FPAØAB CHARAC LBA18  FPAØ FPAØB+1 FPAØ+2 FPAØ+3 STRING TO FLOATING POINT ZERO	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT T THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * * * * * * * * * * * * * * * *
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4240 4241 4242 4244 4244 4245 4246 4250 4251 4250 4251 4252 4253 4256 4257 4258 4258 4258 4258 4258 4258 4258 4258	BCFØ BCF2 BCF4 BCF6 BCF8 BCFC BCFE BDØ2 BDØ4 BDØ6 BDØ9 BDØB BDØD BDØF BD11 BD12 BD14 BD14 BD16 BD18	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53 39  9E 8A 9F 54 9F 4F	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM S DONE FOR THE LOB BCC BSR STB LDA STB SUBA LDA STA JMP STB	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL SER PORTION OF FPAØ WILL RI SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$BØ #\$AØ #\$AØ #\$AØ #\$AB #\$AB FPAØH #\$AB CHARAC LBAIB FPAØH FFAØH FFA	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * * * * * * * * * * * * * * * *
4228 4229 4230 4231 4232 4233 4234 4235 4237 4238 4239 4240 4241 4242 4243 4244 4245 4250 4251 4252 4253 4254 4255 4255 4255 4255 4256 4257 4258 4259 4259 4259 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4261 4262 4263 4260 4261 4260 4261 4261 4261 4261 4261 4261 4261 4261	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFA BDØØ BDØØ BDØØ BDØØ BDØB BDØB BDØB BDØ	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53 39  9E 8A 9F 54 9F 4F 9F 51 9F 52 9F 47	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM S DONE CMPB BCC BSR STB LDA STB SUBA LDA STA LDA STA LDA STB	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL INTEGER PORTION OF FPAØ. AT SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ #\$AØ #\$AB #\$AB #\$AB #\$AB #\$AB #\$AB #\$AB #\$AB	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN FPAØ EXPONENT  * SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * *  * *  (X) = Ø  * ZERO OUT FPAØ & THE SIGN FLAG (COEFCT)  *  * INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO
4228 4228 4230 4231 4232 4233 4234 4235 4237 4238 4237 4240 4241 4242 4243 4244 4245 4245 4250 4251 4250 4251 4252 4253 4254 4255 4256 4257 4258 4257 4258 4257 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4260 4260 4260 4260 4260 4260 4260	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFC BDØØ BDØØ BDØØ BDØØ BDØØ BDØD BDØD BDØ	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53 39 9E 8A 9F 54 9F 54 9F 57 9F 57 9F 47	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM SOONE FITE LDB BCC BSR STB LDA STB LDA STA LDA STA LDA STA LDA STB STB LDA STB LDA STB LDA STB LDA STB LDA STB	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI  FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ FPAØ+3 CHARAC LBA18  FPAØ FPAØ+3 STRING TO FLOATING POINT ZERO FPØSGN FPØSGN FPAØ+3 STRING TO FLOATING POINT ZERO FPØSGN FPØEXP FPAØ+3 FPAØ+4 FPAØ+1 FPAØ+2 FPAØ+6 FPØEXP FYØEXP FYØ	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ  LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * * *  (X) = Ø * ZERO OUT FPAØ & THE SIGN FLAG (COEFCT) * * * INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE RIGHT DECIMAL CTR & DECIMAL PT FLAG TO Ø
4228 4228 4230 4231 4232 4233 4234 4235 4236 4237 4238 4249 4244 4245 4244 4245 4246 4247 4248 4249 4251 4251 4251 4255 4256 4257 4258 4258 4259 4261 4262 4263 4263 4263 4263 4263 4263 4263	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFC BDØØ BDØØ BDØØ BDØØ BDØØ BDØD BDØD BDØ	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53 39  9E 8A 9F 54 9F 4F 9F 51 9F 52 9F 47	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM S DONE CMPB BCC BSR STB LDA STB SUBA LDA STA LDA STA LDA STB	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL INTEGER PORTION OF FPAØ. AT SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ #\$AØ #\$AB #\$AB #\$AB #\$AB #\$AB #\$AB #\$AB #\$AB	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * *  (X) = Ø * ZERO OUT FPAØ & THE SIGN FLAG (COEFCT) * *  * * INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE RIGHT DECIMAL CTR & DECIMAL PT FLAG TO Ø IF CARRY SET (NUMERIC CHARACTER), ASSUME ACCA CONTAINS FIRST
4228 4228 4230 4231 4232 4233 4234 4235 4237 4238 4237 4240 4241 4242 4243 4244 4245 4245 4250 4251 4250 4251 4252 4253 4254 4255 4256 4257 4258 4257 4258 4257 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4261 4260 4260 4260 4260 4260 4260 4260 4260	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BDØØ BDØ9 BDØ9 BDØ8 BDØ9 BDØ8 BDØ9 BDØ8 BDØ9 BDØ8 BDØ9 BDØ8 BDØ9 BDØ9 BDØ11 BD12 BD14 BD12 BD14 BD16 BD18 BD18 BD18 BD18 BD18 BD18 BD18 BD18	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53 39 9E 8A 9F 54 9F 54 9F 57 9F 57 9F 47	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM SOONE FITE LDB BCC BSR STB LDA STB LDA STA LDA STA LDA STA LDA STB STB LDA STB LDA STB LDA STB LDA STB LDA STB	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI  FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ FPAØ+3 CHARAC LBA18  FPAØ FPAØ+3 STRING TO FLOATING POINT ZERO FPØSGN FPØSGN FPAØ+3 STRING TO FLOATING POINT ZERO FPØSGN FPØEXP FPAØ+3 FPAØ+4 FPAØ+1 FPAØ+2 FPAØ+6 FPØEXP FYØEXP FYØ	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ  LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * * *  (X) = Ø * ZERO OUT FPAØ & THE SIGN FLAG (COEFCT) * * * INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE RIGHT DECIMAL CTR & DECIMAL PT FLAG TO Ø
4228 4229 4230 4231 4232 4233 4234 4235 4237 4238 4240 4241 4242 4243 4244 4244 4245 4250 4251 4252 4253 4254 4255 4256 4257 4256 4257 4258 4259 4260 4261 4262 4263 4264 4264 4266 4266 4266 4266	BCFØ BCF2 BCF4 BCF6 BCFB BCFA BCFC BCFE BDØ2 BDØ2 BDØ3 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 50 D7 51 D7 52 D7 53 39 9E 8A 9F 54 9F 4F 9F 51 9F 52 9F 47 9F 45 25 64 BD Ø1 97 81 2D	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM SOUNCE EXTREM SOUNCE EXTREM SOUNCE EXTREM BCC BSC STB LDA STB SUBA LDA STA LDA STA LDA STA LDA STA LDA STB	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL INTEGER PORTION OF FPAØ. AT SER PORTION OF FPAØ WILL RI FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$8Ø #\$AØ #\$AØ #\$AB #\$AB #\$AB #\$AB #\$AB #\$AB  FPAØ #\$AB LBA18  FPAØ #\$AB  FPAØ #\$	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * * * *  (X) = Ø * ZERO OUT FPAØ & THE SIGN FLAG (COEFCT) * * * * INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE RIGHT DECIMAL CTR & DECIMAL PT FLAG TO Ø IF CARRY SET (NUMERIC CHARACTER), ASSUME ACCA CONTAINS FIRST NUMERIC CHAR, SIGN IS POSITIVE AND SKIP THE RAM HOOK
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4249 4241 4242 4243 4244 4245 4246 4247 4251 4251 4253 4253 4255 4255 4256 4257 4258 4258 4259 4261 4262 4263 4266 4267 4266 4267 4268 4269	BCFØ BCF2 BCF4 BCFA BCFA BCFB BCFB BDØ2 BDØ2 BDØ4 BDØ6 BDØ8 BDØ1 BDØ8 BDØ1 BDØ8 BDØ1 BDØ8 BDØ1 BDØ8 BDØ1 BDØ8 BDØ1 BDØ8 BDØ1 BDØ8 BDØ9 BDØ8 BDØ9 BDØ9 BDØ9 BDØ9 BDØ9 BDØ9 BDØ9 BDØ9	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 50 D7 51 D7 52 D7 53 39  9E 8A 9F 54 9F 4F 9F 51 9F 52 9F 47 9F 45 25 64 BD Ø1 97 81 2D 26 Ø4	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	T STATE EXTREM S DONE FOR THE EXTREM B DONE B DONE B B CC B B SR  STB LDA STB	MENT WILL "DENORMALIZE" FP/E E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI  FPØEXP **SAØ LBD11 LBCC8  FPSBYT FPØSGN **SAØ #*SAØ #*SAØ #*PØEXP FPAØ+3 CHARAC LBA18  FPAØ FPAØ+3 CHARAC LBA18  FPAØ FPAØ+1 FPAØ+2 FPAØ+3 STRING TO FLOATING POINT ZERO FPØSGN FPØSGN FPØEXP FPAØ+3  STRING TO FLOATING POINT ZERO FPØSGN FPØEXP FPAØ+1 FPAØ+2 V47 V45 LBD86  RVEC19 #'- LBD2D	A8 - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * *  (X) = Ø * ZERO OUT FPAØ & THE SIGN FLAG (COEFCT)  *  *  INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE RIGHT DECIMAL CTR & DECIMAL PT FLAG TO Ø IF CARRY SET (NUMERIC CHARACTER), ASSUME ACCA CONTAINS FIRST NUMERIC CHAR, SIGN IS POSITIVE AND SKIP THE RAM HOOK HOOK INTO RAM * CHECK FOR A LEADING MINUS SIGN AND BRANCH * IF NO MINUS SIGN
4228 4228 4230 4231 4232 4233 4234 4235 4236 4237 4248 4241 4242 4244 4245 4246 4247 4248 4245 4250 4251 4252 4253 4254 4256 4257 4258 4256 4257 4258 4256 4260 4261 4262 4263 4264 4264 4266 4267 4268 4266 4267 4268 4268 4268 4268 4268 4268 4268 4268	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BD004 BD005 BD005 BD005 BD007 BD011 BD112 BD114 BD116 BD18 BD18 BD18 BD18 BD18 BD18 BD18 BD18	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 5Ø D7 51 D7 52 D7 53 39  9E 8A 9F 54 9F 55 9F 47 9F 52 9F 47 9F 52 9F 47 9F 45 25 64 BD Ø1 97 81 2D 26 Ø4 Ø3 55	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	STATE EXTREM S DONE FOR THE EXTREM B DONE B THE EXTREM B BC BSR  STB LDA STB SUBA LDA STA LDA STA JMP  STB STB STB STB STB STB STB STB STB ST	MENT WILL "DENORMALIZE" FP, E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI  FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN FPØSGN #\$AØ #\$AØ #\$AØ #\$AØ #\$AØ #\$AØ #\$AØ #\$AØ	AØ - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT I THE CONCLUSION OF THE DE-NORMALIZATION MAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN FORCE MANTISSA SIGN ** SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  ** LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB **  **  **  INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE RIGHT DECIMAL CTR & DECIMAL PT FLAG TO Ø IF CARRY SET (NUMERIC CHARACTER), ASSUME ACCA CONTAINS FIRST NUMERIC CHAR, SIGN IS POSITIVE AND SKIP THE RAM HOOK HOOK INTO RAM ** CHECK FOR A LEADING MINUS SIGN AND BRANCH ** IF NOR MINUS SIGN TOGGLE SIGN; Ø = +; FF = -
4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4249 4241 4242 4243 4244 4245 4246 4247 4251 4251 4253 4253 4255 4255 4256 4257 4258 4258 4259 4261 4262 4263 4266 4267 4266 4267 4268 4269	BCFØ BCF2 BCF4 BCF6 BCF8 BCFA BCFC BCFE BDØ2 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5 BDØ5	D6 4F C1 AØ 24 1D 8D D2 D7 63 96 54 D7 54 8Ø 8Ø 86 AØ 97 4F 96 53 97 Ø1 7E BA 18 D7 50 D7 51 D7 52 D7 53 39  9E 8A 9F 54 9F 4F 9F 51 9F 52 9F 47 9F 45 25 64 BD Ø1 97 81 2D 26 Ø4	* THE IN  * TO THE  * THIS IS  * BYTES (  * ONLY TH  INT  *  LBD09  LBD11  * CONVER	T STATE EXTREM S DONE FOR THE EXTREM B DONE B DONE B B CC B B SR  STB LDA STB	MENT WILL "DENORMALIZE" FP/E E RIGHT OF THE MANTISSA TO THE MANTISSA OF FPAØ WILL ( INTEGER PORTION OF FPAØ WILL RI  FPØEXP #\$AØ LBD11 LBCC8  FPSBYT FPØSGN FPØSGN #\$AØ #\$AØ FPAØ+3 CHARAC LBA18  FPAØ FPAØ+3 CHARAC LBA18  FPAØ FPAØ+1 FPAØ+2 FPAØ+3  STRING TO FLOATING POINT ZERO FPØSGN FPØSGN FPØSGN FPAØ+3  CHARAC LBA18  FPAØ FPAØ+1 FPAØ+2 FPAØ+3  TERO FPØSGN FPØEXP FPAØ+1 FPAØ+2 V47 V45 LBD86  RVEC19 #'- LBD2D COEFCT LBD31	A8 - THAT IS IT WILL SHIFT THE BINARY POINT FORCE ITS EXPONENT TO BE \$AO. ONCE CONTAIN THE FOUR LEAST SIGNIFICANT THE CONCLUSION OF THE DE-NORMALIZATION EMAIN.  GET EXPONENT OF FPAØ LARGEST POSSIBLE INTEGER EXPONENT RETURN IF FPAØ >= 32768 SHIFT THE BINARY POINT ONE TO THE RIGHT OF THE LS BYTE OF THE FPAØ MANTISSA ACCB = Ø: ZERO OUT THE SUB BYTE GET MANTISSA SIGN FORCE MANTISSA SIGN TO BE POSITIVE SET CARRY IF MANTISSA * GET DENORMALIZED EXPONENT AND * SAVE IT IN FPAØ EXPONENT = GET LS BYTE OF FPAØ AND = SAVE IT IN CHARAC NORMALIZE FPAØ  * LOAD MANTISSA OF FPAØ WITH CONTENTS OF ACCB * *  (X) = Ø * ZERO OUT FPAØ & THE SIGN FLAG (COEFCT)  *  *  INITIALIZE EXPONENT & EXPONENT SIGN FLAG TO ZERO INITIALIZE RIGHT DECIMAL CTR & DECIMAL PT FLAG TO Ø IF CARRY SET (NUMERIC CHARACTER), ASSUME ACCA CONTAINS FIRST NUMERIC CHAR, SIGN IS POSITIVE AND SKIP THE RAM HOOK HOOK INTO RAM * CHECK FOR A LEADING MINUS SIGN AND BRANCH * IF NO MINUS SIGN

4273	BD2F	26 Ø	и			BNE	LBD35	* IF NOT A PLUS SIGN
4274	BD31	9D 9			LBD31	JSR	GETNCH	GET NEXT INPUT CHARACTER FROM BASIC
4275	BD33	25 5			LDDJI	BCS	LBD86	BRANCH IF NUMERIC CHARACTER
4276	BD35	81 2			LBD35	CMPA		DECIMAL POINT?
4277	BD37	27 2	8			BEQ	LBD61	YES
4278	BD39	81 4	5			CMPA	#'E	"E" SHORTHAND FORM (SCIENTIFIC NOTATION)?
4279	BD3B	26 2	8			BNE	LBD65	NO
4280			_		* EVALUAT		NENT OF EXPONENTIAL FORMAT	
4281	BD3D	9D 9				JSR	GETNCH	GET NEXT INPUT CHARACTER FROM BASIC
4282 4283	BD3F BD41	25 6 81 A				BCS CMPA	LBDA5 #\$AC	BRANCH IF NUMERIC MINUS TOKEN?
4284		27 Ø				BEQ	LBD53	YES
4285	BD45	81 2				CMPA		ASCII MINUS?
4286	BD47	27 Ø	1A			BEQ	LBD53	YES
4287	BD49	81 A	В			CMPA	#\$AB	PLUS TOKEN?
4288	BD4B	27 Ø				BEQ	LBD55	YES
4289		81 2				CMPA		ASCII PLUS?
429Ø 4291	BD4F BD51	27 Ø 20 Ø				BEQ BRA	LBD55 LBD59	YES BRANCH IF NO SIGN FOUND
4292	BD51	Ø3 4			LBD53	COM	V48	SET EXPONENT SIGN FLAG TO NEGATIVE
4293	0000	D5 1						CONVERT IT TO BINARY IN V47
4294	BD55	9D 9	F		LBD55	JSR	GETNCH	GET NEXT INPUT CHARACTER FROM BASIC
4295	BD57	25 4	C			BCS	LBDA5	IF NUMERIC CHARACTER, CONVERT TO BINARY
4296	BD59	ØD 4	8		LBD59	TST	V48	* CHECK EXPONENT SIGN FLAG
4297	BD5B	27 Ø				BEQ	LBD65	* AND BRANCH IF POSITIVE
4298	BD5D	00 4				NEG	V47	NEGATE VALUE OF EXPONENT
4299	BD5F	20 0			10061	BRA	LBD65	*TOGGLE DECIMAL PT FLAG AND INTERPRET ANOTHER
4300 4301	BD61 BD63	Ø3 4			LBD61	COM BNE	V46 LBD31	*TUGGLE DECIMAL PI FLAG AND INTERPRET ANOTHER  *CHARACTER IF <> Ø - TERMINATE INTERPRETATION
4302	5503	20 0	, ,		*	DIVL	LBD31	IF SECOND DECIMAL POINT
43Ø3					* ADJUST	FPAØ FO	OR THE DECIMAL EXPONENT IN	
4304	BD65	96 4	7		LBD65	LDA	V47	* GET EXPONENT, SUBTRACT THE NUMBER OF
43Ø5	BD67	90 4				SUBA	V45	* PLACES TO THE RIGHT OF DECIMAL POINT
4306	BD69	97 4				STA	V47	* AND RESAVE IT.
4307	BD6B	27 1				BEQ	LBD7F	EXIT ROUTINE IF ADJUSTED EXPONENT = ZERO
43Ø8 43Ø9	BD6D BD6F	2A Ø BD B			LBD6F	BPL JSR	LBD78 LBB82	BRANCH IF POSITIVE EXPONENT DIVIDE FPAØ BY 10
4310	BD72	ØC 4			LBUOF	INC	V47	INCREMENT EXPONENT COUNTER (MULTIPLY BY 10)
4311	BD74	26 F				BNE	LBD6F	KEEP MULTIPLYING
4312	BD76	20 0				BRA	LBD7F	EXIT ROUTINE
4313	BD78	BD B	B 6A		LBD78	JSR	LBB6A	MULTIPLY FPAØ BY 10
4314	BD7B	ØA 4	7			DEC	V47	DECREMENT EXPONENT COUNTER (DIVIDE BY 10)
4315	BD7D	26 F				BNE	LBD78	KEEP MULTIPLYING
4316	BD7F	96 5			LBD7F	LDA	COEFCT	GET THE SIGN FLAG
4317	BD81	2A 8	3 E			BPL	LBD11	RETURN IF POSITIVE
	DD03	7 E D	E EO			IMD	IDEEO	TOCCLE MANTICCA CICN OF EDAG IE NECATIVE
4318	BD83	7E B	E E9		*MIII T T D I V	JMP FPAØ F	LBEE9	TOGGLE MANTISSA SIGN OF FPAØ, IF NEGATIVE
4319						FPAØ E	LBEE9 BY TEN AND ADD ACCA TO THE	RESULT
	BD83 BD86 BD88	7E B D6 4 DØ 4	5		*MULTIPLY LBD86	FPAØ E LDB	BY TEN AND ADD ACCA TO THE	
4319 4320	BD86	D6 4	15 16			FPAØ E LDB	BY TEN AND ADD ACCA TO THE V45	RESULT *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT
4319 4320 4321 4322 4323	BD86 BD88 BD8A	D6 4	15 16			FPAØ E LDB SUBB STB	BY TEN AND ADD ACCA TO THE V45 V46	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE
4319 4320 4321 4322 4323 4324	BD86 BD88 BD8A BD8C	D6 4 DØ 4 D7 4	15 16 15		LBD86	FPAØ E LDB SUBB STB	BY TEN AND ADD ACCA TO THE V45 V46 V45	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK
4319 4320 4321 4322 4323 4324 4325	BD86 BD88 BD8A BD8C BD8E	D6 4 DØ 4 D7 4 34 Ø BD B	5  6  5  5  2  B 6A		LBD86	FPAØ E LDB SUBB STB PSHS JSR	BY TEN AND ADD ACCA TO THE V45 V46 V45 A LBB6A	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAYE NEW DIGIT ON STACK MULTIPLY FPAØ BY 10
4319 4320 4321 4322 4323 4324 4325 4326	BD86 BD88 BD8A BD8C BD8E BD91	D6 4 DØ 4 D7 4 34 Ø BD B 35 Ø	15 16 15 12 18 6A		LBD86	FPAØ E LDB SUBB STB PSHS JSR PULS	BY TEN AND ADD ACCA TO THE V45 V46 V45 A LBB6A B	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø GET NEW DIGIT BACK
4319 4320 4321 4322 4323 4324 4325 4326 4327	BD86 BD88 BD8A BD8C BD8E BD91 BD93	D6 4 DØ 4 D7 4 34 Ø BD B 35 Ø CØ 3	15 16 15 12 18 6A		LBD86	FPAØ E LDB SUBB STB PSHS JSR PULS SUBB	BY TEN AND ADD ACCA TO THE V45 V46 V45 A LBB6A B #'0	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 10 GET NEW DIGIT BACK MASK OFF ASCII
4319 4320 4321 4322 4323 4324 4325 4326	BD86 BD88 BD8A BD8C BD8E BD91 BD93	D6 4 DØ 4 D7 4 34 Ø BD B 35 Ø	15 15 18 18 18 18 18 18		LBD86	FPAØ E LDB SUBB STB PSHS JSR PULS	BY TEN AND ADD ACCA TO THE V45 V46 V45 A LBB6A B	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø GET NEW DIGIT BACK
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95	D6 4 DØ 4 D7 4 34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B	15 16 15 12 18 18 14 19 19 19 19 19 19 19 19 19 19 19 19 19		LBD86	FPAØ E LDB SUBB STB PSHS JSR PULS SUBB BSR	SY TEN AND ADD ACCA TO THE V45 V46 V45 A LBB6A B #'0 LBD99	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 10 GET NEW DIGIT BACK MASK OFF ASCII ADD ACCB TO FPAØ GET ANOTHER CHARACTER FROM BASIC PACK FPAØ AND SAVE IT IN FPA3
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD99 BD9C	D6 4 DØ 4 D7 4  34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B BD B	15 16 15 12 18 18 14 18 18 18 18 18 18 18 18 18 18 18 18 18		LBD86 *	PSHS JSR PULS SUBB BSR BRA JSR JSR	Y TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø GET NEW DIGIT BACK MASK OFF ASCII ADD ACCB TO FPAØ GET ANOTHER CHARACTER FROM BASIC PACK FPAØ AND SAVE IT IN FPA3 CONVERT ACCB TO FP NUMBER IN FPAØ
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD99 BD9C BD9F	D6 4 DØ 4 D7 4  34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B BD B 8E Ø	15 16 15 12 18 18 14 18 18 18 18 18 18 18 18 18 18 18 18 18		LBD86 *	PSHS JSR PULS SUBB BSR BRA JSR LDX	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45 A LBB6A B #' Ø LBD99 LBD31 LBC2F LBC7C #V40	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332 4333	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD99 BD9C	D6 4 DØ 4 D7 4  34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B BD B	15 16 15 12 18 18 14 18 18 18 18 18 18 18 18 18 18 18 18 18		* LBD99	PAM E LDB SUBB STB  PSHS JSR PULS SUBB BSR BRA JSR JSR LDX JMP	A LBB6A B #'0 LBD99 LBC2F LBC7C #V49 LB9C2	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø  GET NEW DIGIT BACK MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPAØ  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD99 BD9C BD9F	D6 4 DØ 4 D7 4  34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B BD B 8E Ø	15 16 15 12 18 18 14 18 18 18 18 18 18 18 18 18 18 18 18 18		LBD99 * MULTIPL	PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E	A LBB6A B #'Ø LBD99 LBD31 LBC2F LBC7C #V48 LBC7C #V48 LB T M M M M M M M M M M M M M M M M M M	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø  GET NEW DIGIT BACK MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPAØ  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4328 4330 4331 4332 4333	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD99 BD9C BD9F	D6 4 DØ 4 D7 4 34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B BD B 8E Ø 7E B	15 16 15 18 18 18 18 18 18 18 18 18 18 18 18 18		LBD99 * MULTIPL	PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E	A LBB6A B #'0 LBD99 LBC2F LBC7C #V49 LB9C2	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø  GET NEW DIGIT BACK MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPAØ  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335	BD86 BD88 BD8A BD8C BD91 BD93 BD95 BD97 BD99 BD99 BD99 BD9F BDA2	D6 4 DØ 4 D7 4 34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B BD B BE Ø 7E B	15 16 15 18 18 18 18 18 18 18 18 18 18 18 18 18		LBD99  * MULTIPL * ACCA -	PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 ESAVE BE	BY TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 #V40 LB9C2 HARY RESULT IN V47	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø  GET NEW DIGIT BACK MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPAØ  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4330 4331 4332 4333 4334 4335 4336	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD99 BD9C BD9F BDA2	D6 4 DØ 4 D7 4 34 Ø BD B 35 Ø CØ 3 8D Ø 2Ø 9 BD B BD B BE Ø 7 E B	15 16 15 18 18 18 18 18 18 18 18 18 18 18 18 18		LBD99  * MULTIPL * ACCA -	FPAØ E LDB SUBB STB  PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE B LDB	BY TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 #V40 LB9C2 HARY RESULT IN V47	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * * FPA3  SER IN
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339	BD86 BD88 BD8A BD8C BD8E BD91 BD95 BD97 BD99 BD9C BD9F BDA2	D6 4 4 D7 4 34 0 BD B BD B BD B BE 0 BE 0 F E B BD B BB	15 15 16 16 15 16 16 16 16 16 16 16 16 16 16 16 16 16		LBD99  * MULTIPL * ACCA -	PAME LDB SUBB STB  PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE BI LDB ASLB ASLB ADDB	BY TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LB099 LB031 LBC2F LBC7C #V40 LB972 JENCY AND ADD TO ASCII NUME (NARY RESULT IN V47	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4339 4340	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD96 BD9F BDA2 BDA5 BDA5 BDA7 BDA8 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9	D6 4 4 58 UP 4 4 58	15 16 15 16 16 16 16 16 16 16 16 16 16 16 16 16		LBD99  * MULTIPL * ACCA -	PAME LDB SUBB STB  PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE BILDB ASLB ASLB ASLB ASLB ASLB	BY TEN AND ADD ACCA TO THE V45 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V44 LB9C2 SY 10 AND ADD TO ASCII NUME (NARY RESULT IN V47 V47	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK MULTIPLY FPAØ BY 1Ø GET NEW DIGIT BACK MASK OFF ASCII ADD ACCB TO FPAØ GET ANOTHER CHARACTER FROM BASIC PACK FPAØ AND SAVE IT IN FPA3 CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3 SER IN  TIMES 2 TIMES 4 ADD 1 = TIMES 5 TIMES 1Ø
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4331 4332 4331 4332 4333 4335 4336 4337 4338 4339 4340 4341	BD86 BD88 BD8A BD8C BD8E BD91 BD93 BD95 BD97 BD96 BD9F BDA2 BDA5 BDA7 BDA8 BDA8 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9	D6 4 4 58 BD 8 4 80 3 3 8 8 8 8 8 8 3 3 8 8 8 8 8 8 8 8	15 16 15 17 18 18 18 18 18 18 18 18 18 18 18 18 18		LBD99  * MULTIPL * ACCA -	FPAØ E LDB SUBB STB  PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE B1 LDB ASLB ASLB ASLB ASLB ASLB SUBA	BY TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 ENARY RESULT IN V47 V47 #'0	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SIER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH
4319 4320 4321 4322 4323 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342	BD86 BD88 BD8A BD8C BD91 BD93 BD95 BD97 BD96 BD97 BD96 BD97 BDA2 BDA2	D6 4 4 D7 4 4 BD BD BD BBD BBD BBD BBD BBBD B	155 166 155 188 184 181 182 188 180 180 180 180 180 180 180 180 180		LBD99  * MULTIPL * ACCA -	FPAØ E LDB SUBB SUBB STB PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE B LDB ASLB ASLB ASLB ASLB ASLB ASLB ASLB ASL	BY TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 BY 10 AND ADD TO ASCII NUME NARY RESULT IN V47 V47  #'0 B	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND
4319 4320 4321 4322 4323 4324 4325 4326 4327 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343	BD86 BD88 BD8A BD8C BD91 BD93 BD95 BD97 BD96 BD96 BD96 BDA2	D6 4 4 BD 8 BD 8 BD 8 BB 0 B BB 6 BB 6 BB 6 BB 6 BB 6 BB 6	55 66 55 12 18 18 18 18 18 18 18 18 18 18 18 18 19 19 17 17 17		LBD99  * MULTIPL * ACCA -	FPAB E LDB SUBB SSTB PSHS JSR PULS SUBB BSR BRA JSR JJSR LDX JSR LDX JSR ASLB ASLB ASLB ASLB ASLB ASLB ADDA	BY TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 Y1 10 AND ADD TO ASCII NUME NARY RESULT IN V47 V47  V47  #'0 B B ,S+	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND  ADD 1T TO ACCB
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4330 4331 4331 4332 4333 4334 4335 4337 4338 4337 4338 4344	BD86 BD88 BD8A BD8C BD91 BD93 BD95 BD97 BD96 BD97 BD96 BD97 BDA2 BDA2	D6 4 4 D0 4 4 BD 8 BD 8 BD 8 BD 8 BD 8 BD 8 BBD	55 66 65 55 88 60 60 77 77 77 77 77		LBD99  * MULTIPL * ACCA -	FPAØ E LDB SUBB SUBB STB PSHS JSR PULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE B LDB ASLB ASLB ASLB ASLB ASLB ASLB ASLB ASL	BY TEN AND ADD ACCA TO THE V45 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V44 UB9C2 BY 10 AND ADD TO ASCII NUME INARY RESULT IN V47 V47  V47  #'0 B B, S+ V47	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND
4319 4320 4321 4322 4323 4324 4325 4326 4327 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343	BD86 BD88 BD8A BD8C BD91 BD93 BD95 BD97 BD96 BD97 BD96 BDA2 BDA5 BDA7 BDA8 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9	D6 4 4 D0 4 4 BD 8 BD 8 BD 8 BD 8 BD 8 BD 8 BBD	55 66 65 55 88 60 60 77 77 77 77 77		LBD99  * MULTIPL * ACCA -	FPAØ E LDB SSUBB SSUBB SSUBB PSHS SUBB BSR JJSR JJSR JJSR JJSR JJSR JJSR JJS	BY TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 Y1 10 AND ADD TO ASCII NUME NARY RESULT IN V47 V47  V47  #'0 B B ,S+	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 1Ø GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ GET ANOTHER CHARACTER FROM BASIC PACK FPAØ AND SAVE IT IN FPA3  **CONVERT ACCB TO FP NUMBER IN FPAØ  **ADD FPAØ TO  **FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 1Ø  **MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND ADD IT TO ACCB  SAVE IN V47
4319 4320 4321 4322 4323 4324 4325 4326 4327 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344 4344	BD86 BD88 BD82 BD91 BD93 BD95 BD97 BD96 BD97 BD96 BD96 BD97 BD96 BD96 BD96 BD96 BD48 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9	D6 4 4 D7 4 BD B B B B B B B B B B B B B B B B B	15 15 16 15 15 16 16 16 16 16 16 16 16 16 16 16 16 16	1F FD	LBD99  * MULTIPL * ACCA - LBDA5	FPAØ E LDB STB  PSHS SUBB STB  PSHS SUBB BSR JJSR JJSR JJSR JJSR JJSR JJSR JJS	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LB031 LBC2F LBC7C #V44 LB9C2 SY 10 AND ADD TO ASCII NUME (NARY RESULT IN V47 V47  V47  #'0 B B S S+ V47 LB055 \$9B,\$3E,\$BC,\$1F,\$FD	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 1Ø GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ GET ANOTHER CHARACTER FROM BASIC PACK FPAØ AND SAVE IT IN FPA3  **CONVERT ACCB TO FP NUMBER IN FPAØ  **ADD FPAØ TO  **FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 1Ø  **MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND ADD IT TO ACCB  SAVE IN V47
4319 4320 4321 4322 4323 4324 4325 4326 4327 4329 4330 4331 4332 4333 4334 4335 4336 4335 4336 4337 4348 4349 4341 4342 4343 4344 4345 4346 4347 4348	BD86 BD88 BD86 BD97 BD97 BD99 BD95 BD96 BD96 BD97 BD96 BD96 BDA2	D6 4 4 58	15 15 16 16 15 16 16 16 16 16 16 16 16 16 16 16 16 16	27 FD	LBD99  * MULTIPL * ACCA - LBDA5	FPAØ E LDB STB  PSHS JSR PULS BBSR BRA JSR LDX JJMP LDB ASLB ASLB ADDB ASLB ADDB ASLB BASLB ADDB ASLB ADDB ASLB FCB	8Y TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 SY 10 AND ADD TO ASCII NUME NARY RESULT IN V47 V47  #'0 B ,S+ V47 LBD55  \$98,\$3E,\$BC,\$1F,\$FD \$96,\$6E,\$6E,\$27,\$FD	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FRAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 9999999999  * 9999999999
4319 4320 4321 4322 4323 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349	BD86 BD88 BD86 BD97 BD97 BD99 BD95 BD96 BD96 BD97 BD96 BD96 BDA2	D6 4 4 58	15 15 16 16 15 16 16 16 16 16 16 16 16 16 16 16 16 16		LBD99  * MULTIPL * ACCA - LBDA5	FPAØ E LDB STB  PSHS JSR PULS BBSR BRA JSR LDX JJMP LDB ASLB ASLB ADDB ASLB ADDB ASLB BASLB ADDB ASLB ADDB ASLB FCB	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LB031 LBC2F LBC7C #V44 LB9C2 SY 10 AND ADD TO ASCII NUME (NARY RESULT IN V47 V47  V47  #'0 B B S S+ V47 LB055 \$9B,\$3E,\$BC,\$1F,\$FD	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=Ø, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 1Ø  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPAØ  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 1Ø  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 99999999.9
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4334 4343 4344 4345 4344 4345 4346 4347 4348 4349 4349	BD86 BD88 BD88 BD8E BD91 BD95 BD97 BD96 BD97 BD96 BD97 BD98 BD98 BD98 BD88 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9	D6 4 0 3 4 0 8 0 9 6 0 9 6 0 9 6 0 9 6 0 9 6 0 9 9 6 0 6 0	15 16 6 15 12 12 18 18 18 18 18 18 18 18 18 18 18 18 18	27 FD 28 ØØ	LBD86  *  LBD99  * MULTIPL * ACCA - LBDA5	FPAØ E LDB STB  PSHS SUBB STB  PSHS JSR PULS SUBB BRA JSR BRA JSR BRA JSR BRA JSR BRA JSR BRA BRA FOB ASLB ASLB ASLB ASLB ASLB ASLB ASLB ASL	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #Y40 LB9C2 EV1 10 AND ADD TO ASCII NUME VA7 V47  V47  V47  \$\$\$ B ,S+ V47 LBD55 \$\$\$98,\$38,\$8C,\$1F,\$FD \$\$96,\$66,\$68,\$27,\$FD \$\$95,\$66,\$68,\$28,\$00	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 99999999.9  * 9999999999  * 1E + 09
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4331 4332 4333 4335 4336 4337 4338 4339 4341 4342 4343 4344 4345 4344 4345 4346 4347 4348 4349 4351	BD86 BD88 BD86 BD97 BD97 BD97 BD97 BD97 BD96 BD97 BD96 BD96 BD96 BD96 BD97 BDA2 BDA3 BDA4 BDA4 BDA6 BDA6 BDB8 BDB4 BDB8 BDB8 BDB6 BDB8 BDB6 BDB6 BDB6 BDB6	D6 4 0 34 0 85 0 8 2 0 9 9 6 6 8 8 A 8 A 8 A 8 A 8 A 8 A 8 A 8 A 8	15 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	27 FD 28 ØØ	LBD99  * MULTIPL * ACCA - LBDA5	FPAØ E LDB STB PSHS SUBB SR PPULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE B ASLB ADDB ASLB ADDB ASLB ADDB FCB FCB FCB LDX	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 INARY RESULT IN V47 V47  #'0 B ,S+ V47 LBD55  \$98,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00 #LABE8-1	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 9999999999  * 1E + 09  POINT X TO " IN " MESSAGE
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4334 4343 4344 4345 4344 4345 4346 4347 4348 4349 4349	BD86 BD88 BD88 BD8E BD91 BD95 BD97 BD96 BD97 BD96 BD97 BD98 BD98 BD98 BD88 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9 BDA9	D6 4 4 58 DB 4 58 DB 4 20 9 9 59 6 6 8E 8D 0 8	15 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	27 FD 28 ØØ	LBD86  *  LBD99  * MULTIPL * ACCA - LBDA5	FPAØ E LDB STB PSHS SUBB SR PPULS SUBB BSR BRA JSR LDX JMP Y V47 E SAVE B ASLB ADDB ASLB ADDB ASLB ADDB FCB FCB FCB LDX	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #Y40 LB9C2 EV1 10 AND ADD TO ASCII NUME VA7 V47  V47  V47  \$\$\$ B ,S+ V47 LBD55 \$\$\$98,\$38,\$8C,\$1F,\$FD \$\$96,\$66,\$68,\$27,\$FD \$\$95,\$66,\$68,\$28,\$00	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 99999999.9  * 9999999999  * 1E + 09
4319 4320 4321 4322 4323 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349 4350 4350 4351	BD86 BD88 BD87 BD97 BD97 BD97 BD96 BD97 BD96 BD96 BD96 BD97 BDA5 BDA5 BDA7 BDA8 BDA9 BDA8 BDA9 BDA8 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA9 BDA8 BDA8 BDA8 BDA8 BDA8 BDA8 BDA8 BDA8	D6 4 4 58 DB 4 58 DB 4 20 9 9 59 6 6 8E 8D 0 8	15 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	27 FD 28 ØØ	LBD99  * MULTIPL * ACCA - LBDA5	FPAØ E LDB STB  PSHS SUBB STB  PULS SUBB BSR BRA JSR LDX JMP Y V47 F LDX JMP STAVE BI LDX ASLB ASLB ASLB ASLB ASLB ASLB ASLB ASLB	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V46 V46 V47  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 8Y 10 AND ADD TO ASCII NUME NARY RESULT IN V47 V47  V47  #'0 B , S+ V47 LBD55 \$9B,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00 #LABE8-1 LBDD6	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 99999999.9  * 999999999  * 1E + 09  POINT X TO " IN " MESSAGE COPY A STRING FROM (X) TO CONSOLE OUT GET CURRENT BASIC LINE NUMBER TO ACCD
4319 4320 4321 4322 4323 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349 4350 4350 4350 4350 4351 4352 4353 4354 4365 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375 4375	BD86 BD97 BD97 BD97 BD97 BD96 BD97 BD96 BD96 BD96 BD96 BD80 BD80 BD80 BD80 BD80 BD80 BD80 BD80	D6 4 BD B BE 0 BB BE 0 BB B	15 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	27 FD 28 ØØ	LBD99  * MULTIPL * ACCA - LBDA5   LBD86 LBDB8 LBDC0 * LBDC5  * CONVERT * AND PRI	FPAØ E LDB STB  PSHS JSR PULS SUBBB BRA JSR BRA JSR LDX JMP LDB ASLB ASLB ASLB ASLB ASLB ASLB ASLB ASL	8Y TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 8Y 10 AND ADD TO ASCII NUME NARY RESULT IN V47 V47  W47  #'0 B ,S+ V47 LBD55  \$9B,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00  #LABE8-1 LBDD6 CURLIN IN ACCD INTO A DECIMAL NUM TO CONSOLE OUT	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FRAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD IT TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 999999999  * 1E + 09  POINT X TO " IN " MESSAGE  COPY A STRING FROM (X) TO CONSOLE OUT  GET CURRENT BASIC LINE NUMBER TO ACCD  BBER
4319 4320 4321 4322 4323 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349 4350 4351 4352 4353 4354 4355 4355 4356	BD86 BD88 BD87 BD97 BD97 BD97 BD96 BD97 BD96 BD97 BD96 BD96 BD97 BD96 BD97 BD97 BD96 BD97 BD96 BD97 BD97 BD97 BD97 BD97 BD97 BD97 BD97	D6 4 4 58	15 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	27 FD 28 ØØ	*  LBD99  * MULTIPL * ACCA - LBDA5  *  LBDB6 LBDBB LBDCØ * LBDC5  * CONVERT	FPAØ E LDB STB  PSHS JSR PULS BSR BRA JSR LDX JMP LDX JMP LDX JMP ASLB ASLB ASLB ASLB ASLB ASLB ASLB ASLB	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 SY 10 AND ADD TO ASCII NUME VA7 V47 V47  V47  V47  #'0 B B , S+ V47 LBD55  \$98,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00  #LABEB-1 LBD06 CURLIN IN ACCD INTO A DECIMAL NUM TO CONSOLE OUT FPA0	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE  SAVE NEW DIGIT ON STACK  MULTIPLY FRAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FRAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FRAØ AND SAVE IT IN FRA3  CONVERT ACCB TO FP NUMBER IN FRAØ  * ADD FRAØ TO  * FRA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  **RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 99999999.9  * 1E + 09  POINT X TO " IN " MESSAGE  COPY A STRING FROM (X) TO CONSOLE OUT  GET CURRENT BASIC LINE NUMBER TO ACCD  IBER  SAVE ACCD IN TOP HALF OF FRAØ
4319 4320 4321 4322 4323 4324 4325 4326 4327 4328 4329 4331 4332 4333 4335 4336 4337 4338 4339 4341 4342 4343 4344 4345 4344 4345 4346 4347 4348 4349 4351 4352 4353 4354 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355 4355	BD86 BD88 BD87 BD97 BD97 BD97 BD98 BD96 BD97 BD96 BD96 BD96 BD96 BDA6 BDA6 BDA6 BDA6 BDA6 BDB8 BDB8 BDB8 BDB8 BDB8 BDB8 BDB8 BDB	D6 4 4 5 8 8 4 3 3 4 8 E 8 9 9 E 6 8 8 A 8 D 6 C 6 9 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 9 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0 C 6 0	15 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	27 FD 28 ØØ	LBD99  * MULTIPL * ACCA - LBDA5   LBD86 LBDB8 LBDC0 * LBDC5  * CONVERT * AND PRI	FPAØ E LDB STB  PSHS SUBB BSR BRA JJSR LDX JJSR LDX JMP YVAF E SAVE BI ASLB ASLB ADDB SUBA BRA ASLB ADDB FCB FCB FCB FCB LDX BSR LDX LDX LDB	8Y TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 8Y 10 AND ADD TO ASCII NUME NARY RESULT IN V47 V47  W47  #'0 B ,S+ V47 LBD55  \$9B,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00  #LABE8-1 LBDD6 CURLIN IN ACCD INTO A DECIMAL NUM TO CONSOLE OUT	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FPAØ BY 10 GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPA3  CONVERT ACCB TO FP NUMBER IN FPAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD IT TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 999999999  * 1E + 09  POINT X TO " IN " MESSAGE  COPY A STRING FROM (X) TO CONSOLE OUT GET CURRENT BASIC LINE NUMBER TO ACCD  IBER  SAVE ACCD IN TOP HALF OF FPAØ  REQ D EXPONENT IF TOP HALF OF ACCD = INTEGER
4319 4320 4321 4322 4323 4324 4325 4326 4327 4329 4330 4331 4332 4333 4334 4335 4336 4335 4336 4337 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349 4351 4361 4361 4361 437 4381 4361 437 4381 4385 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4388 4389 4389 4380 4381 4385 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4388 4386 4387 4388 4386 4387 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 43	BD86 BD87 BD97 BD97 BD97 BD96 BD97 BD96 BD97 BD96 BD96 BD96 BD97 BDA2 BDA5 BDA6 BDA6 BDA6 BDA6 BDB8 BDB0 BDB8 BDB6 BDB2 BDB6 BDB6 BDB6 BDB6 BDB6 BDB6	D6 4 4 8 8 8 8 8 8 4 8 8 8 9 8 9 9 6 6 8 8 8 0 0 0 C 6 4 3	15 12 18 18 6A 14 14 16 12 18 18 18 18 18 18 18 18 18 18 18 18 18	27 FD 28 ØØ	LBD99  * MULTIPL * ACCA - LBDA5   LBD86 LBDB8 LBDC0 * LBDC5  * CONVERT * AND PRI	FPAØ E LDB STB  PSHS JSR PPULS BBR BRA JSR LDX JMP LDB ASLB BASL BBASL B	8Y TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 \$Y 10 AND ADD TO ASCII NUME NARY RESULT IN V47  V47  #'0 B ,S+ V47 LBD55  \$98,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00  #LABEB-1 LBDD6 CURLIN IN ACCD INTO A DECIMAL NUM TO CONSOLE OUT FPA0 #*\$90	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FRAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FPAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FPAØ AND SAVE IT IN FPAØ  * ADD FPAØ TO  * FPAØ  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 999999999  * 1E + 09  POINT X TO " IN " MESSAGE  COPY A STRING FROM (X) TO CONSOLE OUT  GET CURRENT BASIC LINE NUMBER TO ACCD  BIBER  SAVE ACCD IN TOP HALF OF FPAØ  REQ D EXPONENT IF TOP HALF OF FACCD = INTEGER  SET CARRY FLAG - FORCE POSITIVE MANTISSA
4319 4320 4321 4322 4323 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349 4350 4351 4352 4353 4354 4355 4356 4357 4356 4357 4358 4359	BD86 BD88 BD87 BD97 BD97 BD97 BD98 BD96 BD97 BD96 BD96 BD96 BD96 BDA6 BDA6 BDA6 BDA6 BDA6 BDB8 BDB8 BDB8 BDB8 BDB8 BDB8 BDB8 BDB	D6 4 4 8 8 8 8 8 8 4 8 8 8 9 8 9 9 6 6 8 8 8 0 0 0 C 6 4 3	15 12 18 18 6A 14 14 16 12 18 18 18 18 18 18 18 18 18 18 18 18 18	27 FD 28 ØØ	LBD99  * MULTIPL * ACCA - LBDA5   LBD86 LBDB8 LBDC0 * LBDC5  * CONVERT * AND PRI	FPAØ E LDB STB  PSHS SUBB BSR BRA JJSR LDX JJSR LDX JMP YVAF E SAVE BI ASLB ASLB ADDB SUBA BRA ASLB ADDB FCB FCB FCB FCB LDX BSR LDX LDX LDB	8Y TEN AND ADD ACCA TO THE V45 V46 V46 V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 SY 10 AND ADD TO ASCII NUME VA7 V47 V47  V47  V47  #'0 B B , S+ V47 LBD55  \$98,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00  #LABEB-1 LBD06 CURLIN IN ACCD INTO A DECIMAL NUM TO CONSOLE OUT FPA0	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FRAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FRAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FRAØ AND SAVE IT IN FRA3  CONVERT ACCB TO FP NUMBER IN FRAØ  * ADD FRAØ TO  * FRA3  SER IN  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD 1T TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 999999999  * 1E + 09  POINT X TO " IN " MESSAGE  COPY A STRING FROM (X) TO CONSOLE OUT  GET CURRENT BASIC LINE NUMBER TO ACCD  IBBER  SAVE ACCD IN TOP HALF OF FRAØ  REQ D EXPONENT IF TOP HALF OF ACCD = INTEGER  SET CARRY FLAG - FORCE POSITIVE MANTISSA  ZERO BOTTOM HALF AND SIGN OF FRAØ, THEN
4319 4320 4321 4322 4323 4324 4325 4326 4327 4329 4330 4331 4332 4333 4334 4335 4336 4335 4336 4337 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349 4351 4361 4361 4361 437 4381 4361 437 4381 4385 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4388 4389 4389 4380 4381 4385 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4386 4387 4388 4386 4387 4388 4386 4387 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 4388 43	BD86 BD87 BD97 BD97 BD97 BD96 BD97 BD96 BD97 BD96 BD96 BD96 BD97 BDA2 BDA5 BDA6 BDA6 BDA6 BDA6 BDB8 BDB0 BDB8 BDB6 BDB2 BDB6 BDB6 BDB6 BDB6 BDB6 BDB6	D6 4 4 8 9 8 4 8 9 7 4 9 9 8 3 6 9 9 6 6 9 9 6 6 9 9 6 6 9 9 6 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9 6 9	15 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	27 FD 28 ØØ	LBD99  * MULTIPL * ACCA - LBDA5   LBD86 LBDB8 LBDC0 * LBDC5  * CONVERT * AND PRI	FPAØ E LDB STB  PSHS JSR PPULS BBR BRA JSR LDX JMP LDB ASLB BASL BBASL B	8Y TEN AND ADD ACCA TO THE V45  A LBB6A B #'0 LBD99 LBD31 LBC2F LBC7C #V40 LB9C2 \$Y 10 AND ADD TO ASCII NUME NARY RESULT IN V47  V47  #'0 B ,S+ V47 LBD55  \$98,\$3E,\$BC,\$1F,\$FD \$9E,\$6E,\$6B,\$27,\$FD \$9E,\$6E,\$6B,\$28,\$00  #LABEB-1 LBDD6 CURLIN IN ACCD INTO A DECIMAL NUM TO CONSOLE OUT FPA0 #*\$90	RESULT  *GET THE RIGHT DECIMAL COUNTER AND SUBTRACT  *THE DECIMAL POINT FLAG FROM IT. IF DECIMAL POINT  *FLAG=0, NOTHING HAPPENS. IF DECIMAL POINT FLAG IS  -1, THEN RIGHT DECIMAL COUNTER IS INCREMENTED BY ONE SAVE NEW DIGIT ON STACK  MULTIPLY FRAØ BY 10  GET NEW DIGIT BACK  MASK OFF ASCII  ADD ACCB TO FRAØ  GET ANOTHER CHARACTER FROM BASIC  PACK FRAØ AND SAVE IT IN FRA3  CONVERT ACCB TO FP NUMBER IN FRAØ  * ADD FPAØ TO  * FPA3  SER IN  TIMES 2  TIMES 2  TIMES 4  ADD 1 = TIMES 5  TIMES 10  *MASK OFF ASCII FROM ACCA, PUSH  *RESULT ONTO THE STACK AND  ADD IT TO ACCB  SAVE IN V47  INTERPRET ANOTHER CHARACTER  * 999999999  * 1E + 09  POINT X TO " IN " MESSAGE  COPY A STRING FROM (X) TO CONSOLE OUT  GET CURRENT BASIC LINE NUMBER TO ACCD  BIBER  SAVE ACCD IN TOP HALF OF FPAØ  REQ D EXPONENT IF TOP HALF OF FPAØ  REQ D EXPONENT IF TOP HALF OF FACCD = INTEGER  SET CARRY FLAG - FORCE POSITIVE MANTISSA

4362	BDD6	7E B9	9 9 C	LBDD6	JMP	LB99C	COPY A STRING FROM (X) TO CONSOLE OUT
4363							
4364				* CONVERT	FP NUM	BER TO ASCII STRING	
4365	BDD9	CE Ø3	B DA	LBDD9	LDU	#STRBUF+3	POINT U TO BUFFER WHICH WILL NOT CAUSE
4366	0000	06.00		*		HCDACE	THE STRING TO BE STORED IN STRING SPACE
4367	BDDC	86 20 D6 54		LBDDC	LDA	#SPACE	SPACE = DEFAULT SIGN FOR POSITIVE #
4368 4369	BDDE BDEØ	2A Ø2			LDB BPL	FPØSGN LBDE4	GET SIGN OF FPAØ BRANCH IF POSITIVE
4309	BDE2	86 21			LDA	#'-	ASCII MINUS SIGN
4371	BDE4	A7 C		LBDE4	STA	,U+	STORE SIGN OF NUMBER
4372	BDE6	DF 64			STU	COEFPT	SAVE BUFFER POINTER
4373	BDE8	97 54	1		STA	FPØSGN	SAVE SIGN (IN ASCII)
4374	BDEA	86 36	3		LDA	#'Ø	ASCII ZERO IF EXPONENT = Ø
4375	BDEC	D6 4F			LDB	FPØEXP	GET FPAØ EXPONENT
4376	BDEE		7 ØØ C6		LBEQ	LBEB8	BRANCH IF FPAØ = Ø
4377 4378	BDF2 BDF3	4F C1 86			CLRA CMPB	#\$80	BASE 10 EXPONENT=0 FOR FP NUMBER > 1 CHECK EXPONENT
4379	BDF5	22 Ø8			BHI	LBDFF	BRANCH IF FP NUMBER > 1
4380	55.0		•	* IF FPAØ			SPEED UP THE CONVERSION PROCESS
4381	BDF7	8E BI	) CØ		LDX	#LBDCØ	POINT X TO FP 1E+Ø9
4382	BDFA	BD BA	A CA		JSR	LBACA	MULTIPLY FPAØ BY (X)
4383	BDFD	86 F7			LDA	#-9	BASE 10 EXPONENT = -9
4384	BDFF	97 45	5	LBDFF	STA	V45	BASE 10 EXPONENT
4385						LIZE THE FP NUMBER TO A V	
4386 4387						RO 99,999,999.9 - THIS IS N WHICH ALL OF THE DIGITS	
4388						ICH CAN BE DISPLAYED WITH	
4389				* SCIENTI			
4390	BEØ1	8E BE	) BB	LBEØ1	LDX	#LBDBB	POINT X TO FP 999,999,999
4391	BEØ4	BD BO	C AØ		JSR	LBCAØ	COMPARE FPAØ TO 999,999,999
4392	BEØ7	2E ØF			BGT	LBE18	BRANCH IF > 999,999,999
4393	BEØ9	8E BI		LBEØ9	LDX	#LBDB6	POINT X TO FP 99,999,999.9
4394	BEØC	BD BC			JSR	LBCAØ	COMPARE FPAØ TO 99,999,999.9
4395 4396	BEØF BE11	2E ØE BD BE			BGT JSR	LBE1F LBB6A	BRANCH IF > 99,999,999.9 (IN RANGE) MULTIPLY FPAØ BY 10
4397	BE14	ØA 45			DEC	V45	SUBTRACT ONE FROM DECIMAL OFFSET
4398	BE16	20 F1			BRA	LBEØ9	PSEUDO - NORMALIZE SOME MORE
4399	BE18	BD BE		LBE18	JSR	LBB82	DIVIDE FPAØ BY 10
4400	BE1B	ØC 45	5		INC	V45	ADD ONE TO BASE 10 EXPONENT
4401	BE1D	20 E2			BRA	LBEØ1	PSEUDO - NORMALIZE SOME MORE
4402	BE1F	BD BS		LBE1F	JSR	LB9B4	ADD .5 TO FPAØ (ROUND OFF)
4403	BE22	BD BC			JSR	LBCC8	CONVERT FPAØ TO AN INTEGER
44Ø4 44Ø5	BE25 BE27	C6 Ø1			LDB LDA	#1 V45	DEFAULT DECIMAL POINT FLAG (FORCE IMMED DECIMAL PT)  * GET BASE 10 EXPONENT AND ADD TEN TO IT
4406	BE29	8B Ø/			ADDA	#9+1	* (NUMBER NORMALIZED TO 9 PLACES & DECIMAL PT)
4407	BE2B	2B Ø9			BMI	LBE36	BRANCH IF NUMBER < 1.0
4408	BE2D	81 ØE			CMPA	#9+2	NINE PLACES MAY BE DISPLAYED WITHOUT
4409				*			USING SCIENTIFIC NOTATION
4410	BE2F	24 Ø5	5		BCC	LBE36	BRANCH IF SCIENTIFIC NOTATION REQUIRED
4411	BE31	4 A			DECA		* SUBTRACT 1 FROM MODIFIED BASE 10 EXPONENT CTR
4412	BE32	1F 89			TFR	A , B	* AND SAVE IT IN ACCB (DECIMAL POINT FLAG)
4413 4414	BE34 BE36	86 Ø2	<u>′</u>	LBE36	LDA DECA	#2	FORCE EXPONENT = Ø - DON'T USE SCIENTIFIC NOTATION * SUBTRACT TWO (WITHOUT AFFECTING CARRY)
4415	BE37	4A		LDLSO	DECA		* FROM BASE 10 EXPONENT
4416	BE38	97 47	,		STA	V47	SAVE EXPONENT - ZERO EXPONENT = DO NOT DISPLAY
4417				*			IN SCIENTIFIC NOTATION
4418	BE3A	D7 45	5		STB	V45	DECIMAL POINT FLAG - NUMBER OF PLACES TO
4419				*			LEFT OF DECIMAL POINT
4420	BE3C	2E Ø[			BGT	LBE4B	BRANCH IF >= 1
4421 4422	BE3E	DE 64 86 28			LDU LDA	COEFPT #'.	POINT U TO THE STRING BUFFER * STORE A PERIOD
4423		A7 C			STA	,U+	* IN THE BUFFER
4424	BE44		•		TSTB	,	CHECK DECIMAL POINT FLAG
4425	BE45	27 Ø4	1		BEQ	LBE4B	BRANCH IF NOTHING TO LEFT OF DECIMAL POINT
4426		86 36			LDA	#'0	* STORE A ZERO
4427	BE49	A7 C	Ĭ		STA	,U+	* IN THE BUFFER
4428							
4429	DEAD	05 01				NTO A STRING OF ASCII DIG	
443Ø 4431		8E BE		LBE4B	LDX LDB	#LBEC5 #Ø+\$8Ø	POINT X TO FP POWER OF 10 MANTISSA INITIALIZE DIGIT COUNTER TO 0+\$80
4432	DL4L	CO OR	,	* BIT 7 S		ISED TO INDICATE THAT THE I	
4433						HEN YOU 'ADD' A NEGATIVE I	
4434						BTRACTING A POSITIVE ONE	
4435				* THE ROU	TINE KN	OWS THAT A 'SUBTRACTION'	IS OCCURING.
4436		96 53		LBE5Ø	LDA	FPAØ+3	* ADD MANTISSA LS
4437		AB Ø3			ADDA	,	* BYTE OF FPAØ
	BE54					FPAØ+3	* AND (X)
4439 4440	BE56 BE58				LDA ADCA	FPAØ+2 2 Y	= ADD MANTISSA = NUMBER 3 BYTE OF
4440		97 52			STA	FPAØ+2	= FPAØ AND (X)
4442		96 51			LDA	FPAØ+1	* ADD MANTISSA
	BE5E				ADCA		* NUMBER 2 BYTE OF
4444	BE6Ø				STA	FPAØ+1	* FPAØ AND (X)
4445		96 50			LDA	FPAØ	= ADD MANTISSA
4446		A9 84			ADCA		= MS BYTE OF
4447		97 50	9		STA	FPAØ	= FPAØ AND (X)
4448	BE68	50			INCB		ADD ONE TO DIGIT COUNTER
	REEG	56					
4449 4450	BE69 BE6A				RORB ROLB		ROTATE CARRY INTO BIT 7 *SET OVERFLOW FLAG AND BRANCH IF CARRY = 1 AND

4451	BE6B	28 E	3			BVC	LBE50	*POSITIVE MANTISSA OR CARRY = Ø AND NEG MANTISSA
4452	BE6D	24 Ø				BCC	LBE72	BRANCH IF NEGATIVE MANTISSA
4453	BE6F	CØ Ø				SUBB	#10+1	* TAKE THE 9 S COMPLEMENT IF
4454	BE71	50				NEGB	#1 <b>0</b> .1	* ADDING MANTISSA
4455	BE72	CB 2	F		LBE72	ADDB	#'0-1	ADD ASCII OFFSET TO DIGIT
4456	BE74	30 0			LULIL	LEAX	4,X	MOVE TO NEXT POWER OF 10 MANTISSA
4457	BE76	1F 9				TFR	B,A	SAVE DIGIT IN ACCA
4458	BE78	84 7				ANDA	#\$7F	MASK OFF BIT 7 (ADD/SUBTRACT FLAG)
4459	BE7A	A7 C				STA		STORE DIGIT IN STRING BUFFER
4460	BE7C	ØA 4				DEC	,U+ V45	DECREMENT DECIMAL POINT FLAG
		26 Ø				BNE		
4461	BE7E						LBE84	BRANCH IF NOT TIME FOR DECIMAL POINT
4462 4463	BE8Ø	86 2 A7 C				LDA	#'.	* STORE DECIMAL POINT IN
	BE82		Ю		1.0504	STA	, U+	* STRING BUFFER
4464	BE84	53	a		LBE84	COMB	##00	TOGGLE BIT 7 (ADD/SUBTRACT FLAG)
4465	BE85 BE87	C4 8		0		ANDB	#\$80 #LBEC5+9*4	MASK OFF ALL BUT ADD/SUBTRACT FLAG
4466		8C B		9		CMPX		COMPARE X TO END OF MANTISSA TABLE
4467	BE8A	26 C	4			BNE	LBE50	BRANCH IF NOT AT END OF TABLE
4468	DE00	46.0	_				ZEROS AND STORE EXPONENT	
4469	BE8C	A6 C			LBE8C	LDA	, -U	GET THE LAST CHARACTER; MOVE POINTER BACK
4470	BE8E	81 3				CMPA	#'0	WAS IT A ZERO?
4471	BE9Ø	27 F				BEQ	LBE8C	IGNORE TRAILING ZEROS IF SO
4472	BE92	81 2				CMPA	#'.	CHECK FOR DECIMAL POINT
4473	BE94	26 Ø				BNE	LBE98	BRANCH IF NOT DECIMAL POINT
4474	BE96	33 5				LEAU	-1,U	STEP OVER THE DECIMAL POINT
4475	BE98	86 2			LBE98	LDA	#'+	ASCII PLUS SIGN
4476	BE9A	D6 4				LDB	V47	GET SCIENTIFIC NOTATION EXPONENT
4477	BE9C	27 1				BEQ	LBEBA	BRANCH IF NOT SCIENTIFIC NOTATION
4478	BE9E	2A Ø				BPL	LBEA3	BRANCH IF POSITIVE EXPONENT
4479	BEAØ	86 2	D			LDA	#'-	ASCII MINUS SIGN
4480	BEA2	50	_			NEGB		NEGATE EXPONENT IF NEGATIVE
4481	BEA3	A7 4			LBEA3	STA	2,U	STORE EXPONENT SIGN IN STRING
4482	BEA5	86 4				LDA	#'E	* GET ASCII E (SCIENTIFIC NOTATION
4483	BEA7	A7 4	1			STA	1,U	* FLAG) AND SAVE IT IN THE STRING
4484	BEA9	86 2	F			LDA	#'0-1	INITIALIZE ACCA TO ASCII ZERO
4485							VALUE IN ACCB TO DECIMAL	
4486					* ASCII N	UMBER (	< 100) IN ACCD	
4487	BEAB	4 C			LBEAB	INCA		ADD ONE TO 10 S DIGIT OF EXPONENT
4488	BEAC	CØ Ø	Α			SUBB	#10	SUBTRACT 10 FROM ACCB
4489	BEAE	24 F	В			BCC	LBEAB	ADD 1 TO 10 S DIGIT IF NO CARRY
4490	BEBØ	CB 3	Α			ADDB	#'9+1	CONVERT UNITS DIGIT TO ASCII
4491	BEB2	ED 4	3			STD	3,U	SAVE EXPONENT IN STRING
4492	BEB4	6F 4	5			CLR	5,U	CLEAR LAST BYTE (TERMINATOR)
4493	BEB6	2Ø Ø	4			BRA	LBEBC	GO RESET POINTER
4494	BEB8	A7 C	4		LBEB8	STA	, U	STORE LAST CHARACTER
4495	BEBA	6F 4	1		LBEBA	CLR	1,U	CLEAR LAST BYTE (TERMINATOR - REQUIRED BY
4496					*			PRINT SUBROUTINES)
4497	BEBC	8E Ø	3 D	A	LBEBC	LDX	#STRBUF+3	RESET POINTER TO START OF BUFFER
4498	BEBF	39				RTS		
4499					*			
4500	BECØ	8Ø Ø	Ø Ø	0 00 00	LBECØ	FCB	\$80,\$00,\$00,\$00,\$00	FLOATING POINT .5
4501					*			
4502							ORMALIZED POWERS OF 10	
4503	BEC5	FA Ø			LBEC5	FCB	\$FA,\$ØA,\$1F,\$ØØ	-100000000
4504	BEC9	ØØ 9		6 80	LBEC9	FCB	\$00,\$98,\$96,\$80	1000000
45Ø5					LBECD	FCB	\$FF,\$FØ,\$BD,\$CØ	-1000000
	BECD	FF F	ØВ		LDLUD		\$00,\$01,\$86,\$A0	
4506	BECD BED1		ØВ		LBED1	FCB		100000
45Ø6 45Ø7		FF F	Ø B 1 8	6 AØ		FCB FCB	\$FF,\$FF,\$D8,\$FØ	100000 -10000
	BED1	FF F ØØ Ø	Ø B 1 8 F D	6 AØ 8 FØ	LBED1		\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$03,\$E8	
4507	BED1 BED5	FF F ØØ Ø FF F	Ø B 1 8 F D Ø Ø	6 AØ 8 FØ 3 E8	LBED1 LBED5	FCB	\$FF,\$FF,\$D8,\$FØ	-10000
45Ø7 45Ø8	BED1 BED5 BED9	FF F ØØ Ø FF F ØØ Ø	Ø B 1 8 F D Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C	LBED1 LBED5 LBED9	FCB FCB	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$03,\$E8	-10000 1000
4507 4508 4509	BED1 BED5 BED9 BEDD	FF F ØØ Ø FF F ØØ Ø FF F	Ø B 1 8 F D Ø Ø F F Ø Ø	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD	FCB FCB FCB	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$9C	-10000 1000 -100
4507 4508 4509 4510 4511 4512	BED1 BED5 BED9 BEDD BEE1	FF F ØØ Ø FF F ØØ Ø FF F	Ø B 1 8 F D Ø Ø F F Ø Ø	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5	FCB FCB FCB	\$FF,\$FF,\$D8,\$FØ \$ØØ,\$ØØ,\$Ø3,\$E8 \$FF,\$FF,\$FF,\$9C \$ØØ,\$ØØ,\$ØØ,\$ØA	-10000 1000 -100 10
4507 4508 4509 4510 4511 4512 4513	BED1 BED5 BED9 BEDD BEE1 BEE5	FF F ØØ Ø FF F ØØ Ø FF F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 *	FCB FCB FCB FCB	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF	-10000 1000 -100 10 -1
4507 4508 4509 4510 4511 4512 4513 4514	BED1 BED5 BED9 BEDD BEE1 BEE5	FF F ØØ Ø FF F ØØ Ø FF F F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5	FCB FCB FCB FCB	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF	-10000 1000 -100 10 -1
4507 4508 4509 4510 4511 4512 4513 4514 4515	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB	FF F F F F F F F F F F F F F F F F F F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 *	FCB FCB FCB FCB FCB	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF	-10000 1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0
4507 4508 4509 4510 4511 4512 4513 4514	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB	FF F ØØ Ø FF F ØØ Ø FF F F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 *	FCB FCB FCB FCB	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF	-10000 1000 -100 10 -1
4507 4508 4509 4510 4511 4512 4513 4514 4515	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB	FF F F F F F F F F F F F F F F F F F F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 *	FCB FCB FCB FCB FCB	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF	-10000 1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB BEED	FF F F F F F F F F F F F F F F F F F F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9	FCB FCB FCB FCB LDA BEQ COM RTS	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF	-10000 1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB BEED	FF F F F F F F F F F F F F F F F F F F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBED0 LBEE1 LBEE5 * * LBEE9	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5	\$FF,\$FF,\$D8,\$FØ \$ØØ,\$ØØ,\$Ø3,\$E8 \$FF,\$FF,\$FF,\$FC \$ØØ,\$ØØ,\$ØØ,\$ØA \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSGN  OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ	-10000 1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB BEED	FF F F F F F F F F F F F F F F F F F F	Ø B 1 8 F D Ø Ø F F Ø Ø F F	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND A-4-BQ-4-* * AND THE	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00,\$0A \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF	-10000 1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB BEED BEEF	FF F F ØØ Ø Ø FF F F ØØ Ø Ø FF F F ØØ Ø Ø FF F F F F F F F F F F F F F F F F F F	Ø B B 1 8 8 F P P P P P P P P P P P P P P P P P	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND A-4-BQ-4-* * AND THE	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØEXP UMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,O	-10000 -1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB BEED BEEF	FF F F ØØ Ø Ø FF F F ØØ Ø Ø FF F F ØØ Ø Ø FF F F F F F F F F F F F F F F F F F F	Ø B B 1 8 8 F P P P P P P P P P P P P P P P P P	6 AØ 8 FØ 3 E8 F 9C Ø ØA	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND A-4-BQ-4-* * AND THE	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00,\$0A \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF	-10000 1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4521	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE8 BEED BEEF	FF F F ØØ Ø Ø FF F F ØØ Ø Ø FF F F F F	Ø B 8 F F P P P P P P P P P P P P P P P P P	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * LBEE9  LBEEF * EXPAND * * AQ+BQ*** * AND THE * COEFFIC	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØEXP UMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,O	-10000 -1000 -100 10 -1 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4521 4522	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB BEED BEEF	FF F F ØØ Ø Ø FF F F ØØ Ø Ø FF F F F F	Ø B 8 F F P P P P P P P P P P P P P P P P P	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * LBEE9  LBEEF * EXPAND * * AQ+BQ*** * AND THE * COEFFIC	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI IENTS A	\$FF,\$FF,\$D8,\$FØ \$ØØ,\$ØØ,\$Ø3,\$E8 \$FF,\$FF,\$FF,\$FC \$ØØ,\$ØØ,\$ØØ,\$ØA \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ***7 WHERE Q = FPAØ \$B,C,D COEFPT	-10000 -1000 -100 -100 -100 -100 -100 -
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4522 4523	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE8 BEED BEEF	FF F F ØØ Ø Ø FF F F ØØ Ø Ø Ø FF F F ØØ Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø	Ø B 8 B F F Ø Ø F F F F 2 4 4 C 2 5	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * LBEE9  LBEEF * EXPAND * * AQ+BQ*** * AND THE * COEFFIC	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI IENTS A STX JSR	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00,\$0A \$FF,\$FF,\$FF,\$FF  FP0EXP FP0EXP FP0SCN  OMIAL OF THE FORM +D0**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D CDEFPT LBC2F	-10000 1000 -100 10 -1  GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0  SAVE COEFFICIENT TABLE POINTER MOVE FPA0 TO FPA3
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4521 4522 4523 4524	BED1 BED5 BED9 BEDD BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7	FF F F ØØ Ø Ø FF F F ØØ Ø Ø Ø FF F F ØØ Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * LBEE9  LBEEF * EXPAND * * AQ+BQ*** * AND THE * COEFFIC	FCB FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI IENTS A STX JSR BSR	\$FF,\$FF,\$D8,\$FØ \$80,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$FF  \$00,\$00,\$00,\$00,\$0A \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSGN  OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBC2F LBEFC	-10000 1000 -100 10 -1  GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0  SAVE COEFFICIENT TABLE POINTER MOVE FPA0 TO FPA3 MULTIPLY FPA3 BY FPA0
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4519 4520 4521 4522 4523 4524 4524 4525	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F ØØ Ø Ø FF F F ØØ Ø Ø FF F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * LBEE9  LBEEF * EXPAND * * AQ+BQ*** * AND THE * COEFFIC	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI IENTS A STX JSR BSR BSR	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSSN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBEFC LBFF01	-10000 -1000 -1000 -100 -100 -100 -100
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4520 4522 4523 4524 4525 4525	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F F F F F F F F F F F F F F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * AQ+BQ*** * AND THE * COEFFIC LBEFØ	FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLLYN 3+CQ**5 X REGI IENTS A STX JSR BSR BSR LDX	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBEFC LBF01 #V40	-10000 1000 -100 10 -1  GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0  SAVE COEFFICIENT TABLE POINTER MOVE FPA0 TO FPA3 MULTIPLY FPA3 BY FPA0 EXPAND POLYNOMIAL POINT X TO FPA3
4507 4508 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4520 4522 4523 4524 4525 4526 4527	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F F F F F F F F F F F F F F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBED1 LBEE1 LBEE5  * LBEE9  LBEEF * EXPAND * * AVBQ** * COEFFIC LBEFØ	FCB FCB FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI IENTS A STX JSR BSR BSR LDX	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$PC \$00,\$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBEFC LBF01 #V40	-10000 -1000 -1000 -1000 -100 -100 -100
4507 4508 4509 4510 4511 4511 4513 4514 4515 4516 4517 4518 4519 4520 4521 4522 4523 4524 4525 4526 4527 4528	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F F F F F F F F F F F F F F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * AQ+BQ** * AND THE * COEFFIC LBEFØ  LBEFC * CALCULA*	FCB FCB FCB FCB FCB FCB LDA BEQ COM RTS 3 + POLYN 3 + CQ**5 X REGI IENTS A STX JSR BSR BSR LDX JMP	\$FF,\$FF,\$D8,\$FØ \$00,\$00,\$00,\$03,\$E8 \$FF,\$FF,\$FF,\$FF \$00,\$00,\$00,\$00,\$00 \$FF,\$FF,\$FF,\$FF  \$FPØEXP LBEEF \$FPØSGN  OMIAL OF THE FORM \$+DQ**7 WHERE Q = FPAØ \$TER POINTS TO A TABLE OF \$,B,C,D COEFPT LBC2F LBECC LBFØ1 #V40 LBACA	-10000 1000 -100 10 -1  GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0  SAVE COEFFICIENT TABLE POINTER MOVE FPA0 TO FPA3 MULTIPLY FPA3 BY FPA0 EXPAND POLYNOMIAL POINT X TO FPA3 MULTIPLY (X) BY FPA0  OMIAL
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4520 4521 4522 4523 4524 4525 4526 4527 4528 4529	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F F F F F F F F F F F F F F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBED1 LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND A: * AND THE * COEFFIC LBEFØ  LBEFC * CALCULA' * EXPRESS * OF COEF	FCB FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI IENTS A STX JSR BSR BSR LDX JMP	SFF, SFF, SD8, SFØ \$00, \$00, \$00, \$03, \$58 \$FF, SFF, SFF, SPC \$00, \$00, \$00, \$00, \$00 \$6F, SFF, SFF, SFF  FPØEXP LBEEF FPØSGN  OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBEFC LBF01 #V40 LBACA  VALUE OF AN EXPANDED POLYN TER WITH (X) POINTING TO A S, THE FIRST BYTE OF WHICH	-10000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4521 4522 4523 4524 4525 4524 4525 4527 4528 4529 4529 4529 4529	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F F F F F F F F F F F F F F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBED1 LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND A: * AND THE * COEFFIC LBEFØ  LBEFC * CALCULA' * EXPRESS * OF COEF	FCB FCB FCB FCB FCB FCB LDA BEQ COM RTS A POLYN 3+CQ**5 X REGI IENTS A STX JSR BSR BSR LDX JMP	SFF, SFF, SD8, SFØ SØØ, SØØ, SØØ, SØ3, SE8 SFF, SFF, SFF, SPC SØØ, SØØ, SØØ, SØA SFF, SFF, SFF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBFFC LBFØ1 #V4Ø LBACA VALUE OF AN EXPANDED POLYN TER WITH (X) POINTING TO A	-10000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4520 4521 4522 4523 4524 4524 4525 4526 4527 4528 4529 4529 4520 4521 4524 4525 4526 4527 4528 4529 4530 4530 4530 4530 4530 4530 4530 4530	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F F F F F F F F F F F F F F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * AQ+B(**: * AND THE * COEFFIC LBEF0  LBEFC * CALCULA' * EXPRESS * OF COEF * NUMBER * OF PACK	FCB FCB FCB FCB FCB FCB  LDA BEQ COM RTS A POLYNN STS A POLYNN STS BSR BSR BSR LDX JMP LDX JMP FICIENTS FOF FCF FCF FCF FCF FCF FCF FCF FCF FCF	SFF, SFF, SD8, SFØ SØØ, SØØ, SØØ, SØ3, SE8 SFF, SFF, SFF, SPC SØØ, SØØ, SØØ, SØA SFF, SFF, SFF  FPØEXP FPØEXP FPØSGN  OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBEG2F LBEFC LBFØ1 #V4Ø LBACA  VALUE OF AN EXPANDED POLYN TER WITH (X) POINTING TO A S, THE FIRST BYTE OF WHICH FFICIENTS-1) FOLLOWED BY T TING POINT NUMBERS. THE	-10000 1000 -1000 10 -10 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0  SAVE COEFFICIENT TABLE POINTER MOVE FPA0 TO FPA3 MULTIPLY FPA3 BY FPA0 EXPAND POLYNOMIAL POINT X TO FPA3 MULTIPLY (X) BY FPA0  OMIAL TABLE IS THE HAT NUMBER
4507 4508 4509 4510 4511 4511 4512 4513 4514 4515 4516 4517 4518 4519 4521 4522 4523 4524 4525 4526 4525 4526 4529 4530 4531 4532	BED1 BED5 BED9 BEE1 BEE5 BEE9 BEEB BEED BEEF BEF7 BEF9	FF F F F F F F F F F F F F F F F F F F	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * AQ+B(**: * AND THE * COEFFIC LBEF0  LBEFC * CALCULA' * EXPRESS * OF COEF * NUMBER * OF PACK	FCB FCB FCB FCB FCB FCB  LDA BEQ COM RTS A POLYNN STS A POLYNN STS BSR BSR BSR LDX JMP LDX JMP FICIENTS FOF FCF FCF FCF FCF FCF FCF FCF FCF FCF	SFF, SFF, SD8, SFØ \$00, \$00, \$00, \$03, \$58 \$FF, SFF, SFF, SPC \$00, \$00, \$00, \$00, \$00 \$FF, SFF, SFF, SFF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBEFC LBFØ1 #V40 LBACA VALUE OF AN EXPANDED POLYN TER WITH (X) POINTING TO A S, THE FIRST BYTE OF WHICH FFFICIENTS-1) FOLLOWED BY T	-10000 1000 -1000 10 -10 GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0  SAVE COEFFICIENT TABLE POINTER MOVE FPA0 TO FPA3 MULTIPLY FPA3 BY FPA0 EXPAND POLYNOMIAL POINT X TO FPA3 MULTIPLY (X) BY FPA0  OMIAL TABLE IS THE HAT NUMBER
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4520 4521 4522 4523 4524 4524 4525 4526 4527 4528 4529 4529 4520 4521 4524 4525 4526 4527 4528 4529 4530 4530 4530 4530 4530 4530 4530 4530	BED1 BED9 BED9 BED9 BEE1 BEE5 BEE9 BEEB BEF0 BEF7 BEF7 BEF7 BEF7	FF F 600 00 FF F F 600 00 00 FF F F 600 00 00 00 00 00 00 00 00 00 00 00 00	Ø B B B B B B B B B B B B B B B B B B B	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * A AQ+BQ*** * AND THE * COEFFIC LBEFØ  LBEFC  * CALCULA' * EXPRESS * OF COEF * NUMBERR * OF PACK * POLYNOM	FCB	SFF, SFF, SD8, SFØ SØØ, SØØ, SØØ, SØ3, SE8 SFF, SFF, SFF, SPC SØØ, SØØ, SØØ, SØA SFF, SFF, SFF  FPØEXP FPØEXP FPØSGN  OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBEG2F LBEFC LBFØ1 #V4Ø LBACA  VALUE OF AN EXPANDED POLYN TER WITH (X) POINTING TO A S, THE FIRST BYTE OF WHICH FFICIENTS-1) FOLLOWED BY T TING POINT NUMBERS. THE	-10000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4520 4521 4522 4523 4524 4522 4523 4524 4525 4526 4527 4528 4529 4530 4531 4532 4533 4534 4534 4535 4536	BED1 BED5 BED9 BEE5 BEE5 BEE5 BEE6 BEE7 BEF7 BEF7 BEF7 BEF7	FF F F F F F F F F F F F F F F F F F F	Ø 1 8 8 8 9 8 9 8 9 9 9 9 9 9 9 9 9 9 9 9	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * AQ+BQ**: * AND THE * COEFFIC LBEFØ  LBEFC * CALCULA' * EXPRESS * OF COEF * NUMBER * OF PACK * POLYNOM * (((FPAØ)*) LBEFF	FCB FCB FCB FCB FCB FCB  LDA EQ	SFF, SFF, SD8, SFØ SØØ, SØØ, SØØ, SØØ, SØØ, SØØ, SØØ, SØØ	-10000 1000 -100 10 -10  GET EXPONENT OF FPA0 BRANCH IF FPA0 = 0 TOGGLE MANTISSA SIGN OF FPA0  SAVE COEFFICIENT TABLE POINTER MOVE FPA0 TO FPA3 MULTIPLY FPA3 BY FPA0 EXPAND POLYNOMIAL POINT X TO FPA3 MULTIPLY (X) BY FPA0  OMIAL TABLE IS THE HAT NUMBER E =  SAVE COEFFICIENT TABLE POINTER
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4521 4522 4523 4524 4525 4526 4527 4528 4529 4530 4531 4533 4534 4535 4535 4535 4535 4535	BED1 BED5 BED9 BEED1 BEE5 BEE8 BEEB BEEF6 BEF7 BEF7 BEF7 BEF7 BEF7	FF F F F F F F F F F F F F F F F F F F	Ø 1 8 8 8 9 8 9 8 9 9 9 9 9 9 9 9 9 9 9 9	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBED0 LBEE1 LBEE5 * * * LBEE9  LBEEF * EXPAND * AQ+BQ**: * AND THE * COEFFIC LBEFØ  LBEFC * CALCULA' * EXPRESS * OF COEF * NUMBER * OF PACK * POLYNOM * (((FPAØ)	FCB	SFF, SFF, SD8, SFØ SØØ, SØØ, SØØ, SØ3, SEB SFF, SFF, SFF, SPC SØØ, SØØ, SØØ, SØA SFF, SFF, SFF, SFF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBFG LBFØ1 #V4Ø LBACA VALUE OF AN EXPANDED POLYN TER WITH (X) POINTING TO A S, THE FIRST BYTE OF WHICH FFICIENTS-1) FOLLOWED BY T TING POINT NUMBERS. THE EVALUATED AS FOLLOWS: VALU *FPAØ+Y2)*FPAØ YN) COEFPT LBC2A	-10000 1000 -1000 1000 -1000 -1000 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4521 4522 4523 4524 4525 4524 4527 4528 4529 4531 4532 4533 4534 4535 4536 4535 4536 4537 4538	BED1 BED5 BED9 BED9 BEE5 BEE5 BEE6 BEF6 BEF7 BEF7 BEF7 BEF7 BEF7 BEF7 BEF7 BEF7	FF F 60 0 0 0 FF F F 60 0 0 0 0 0 FF F F 60 0 0 0	Ø 1 8 8 8 9 8 9 8 9 9 9 9 9 9 9 9 9 9 9 9	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * AQ+BQ**: * AND THE * COEFFIC LBEFØ  LBEFC * CALCULA' * EXPRESS * OF COEF * NUMBER * OF PACK * POLYNOM * (((FPAØ)*) LBEFF	FCB	SFF, SFF, SD8, SFØ \$00, \$00, \$00, \$03, \$58 \$FF, SFF, SFF, SPC \$00, \$00, \$00, \$00, \$00 \$00, \$00, \$00,	-10000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000 -1000
4507 4508 4509 4510 4511 4512 4513 4514 4515 4516 4517 4518 4519 4520 4521 4522 4523 4524 4525 4526 4527 4528 4529 4530 4531 4533 4534 4535 4535 4535 4535 4535	BED1 BED5 BED9 BED9 BEE5 BEE5 BEE6 BEF6 BEF7 BEF7 BEF7 BEF7 BEF7 BEF7 BEF7 BEF7	FF F F F F F F F F F F F F F F F F F F	Ø 1 8 8 8 9 8 9 8 9 9 9 9 9 9 9 9 9 9 9 9	6 AØ 8 FØ 3 E8 F 9C Ø ØA F FF	LBED1 LBED5 LBED9 LBEDD LBEE1 LBEE5 * * LBEE9  LBEEF * EXPAND * AQ+BQ**: * AND THE * COEFFIC LBEFØ  LBEFC * CALCULA' * EXPRESS * OF COEF * NUMBER * OF PACK * POLYNOM * (((FPAØ)*) LBEFF	FCB	SFF, SFF, SD8, SFØ SØØ, SØØ, SØØ, SØ3, SEB SFF, SFF, SFF, SPC SØØ, SØØ, SØØ, SØA SFF, SFF, SFF, SFF  FPØEXP LBEEF FPØSGN OMIAL OF THE FORM +DQ**7 WHERE Q = FPAØ STER POINTS TO A TABLE OF ,B,C,D COEFPT LBC2F LBFG LBFØ1 #V4Ø LBACA VALUE OF AN EXPANDED POLYN TER WITH (X) POINTING TO A S, THE FIRST BYTE OF WHICH FFICIENTS-1) FOLLOWED BY T TING POINT NUMBERS. THE EVALUATED AS FOLLOWS: VALU *FPAØ+Y2)*FPAØ YN) COEFPT LBC2A	-10000 1000 -1000 1000 -1000 -1000 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -100 -

4540	BFØ8	D7 55	i		STB	COEFCT	* USE AND STORE IT IN TEMPORARY COUNTER
4541	BFØA	9F 64			STX	COEFPT	SAVE NEW COEFFICIENT POINTER
4542	BFØC	8D EE		LBFØC	BSR	LBEFC	MULTIPLY (X) BY FPAØ
4543	BFØE	9E 64			LDX	COEFPT	*GET COEFFICIENT POINTER
4544	BF1Ø	30 05			LEAX		*MOVE TO NEXT FP NUMBER
4545	BF12	9F 64			STX	COEFPT	*SAVE NEW COEFFICIENT POINTER
4546	BF14	BD B9	C2		JSR	LB9C2	ADD (X) AND FPAØ
4547	BF17	8E ØØ	45		LDX	#V45	POINT (X) TO FPA4
4548	BF1A	ØA 55	i		DEC	COEFCT	DECREMENT TEMP COUNTER
4549	BF1C	26 EE			BNE	LBFØC	BRANCH IF MORE COEFFICIENTS LEFT
4550	BF1E	39			RTS		
4551							
4552				* RND			
4553	BF1F	BD BC		RND	JSR	LBC6D	TEST FPAØ
4554		2B 21			BMI	LBF45	BRANCH IF FPAØ = NEGATIVE
4555		27 15			BEQ	LBF3B	BRANCH IF FPAØ = Ø
4556		8D 10			BSR	LBF38	CONVERT FPAØ TO AN INTEGER
4557	BF28	BD BC			JSR	LBC2F	PACK FPAØ TO FPA3
4558		8D ØE			BSR	LBF3B	GET A RANDOM NUMBER: FPAØ < 1.0
4559		8E ØØ			LDX	#V4Ø	POINT (X) TO FPA3
4560	BF3Ø	8D CA 8E BA			BSR LDX	LBEFC #LBAC5	MULTIPLY (X) BY FPAØ
4561 4562	BF35	BD B9			JSR	LB9C2	POINT (X) TO FP VALUE OF 1.0 ADD 1.0 TO FPA0
4563		7E BC		LBF38	JMP	INT	CONVERT FPAØ TO AN INTEGER
4564	D1 30	, , ,				ANDOM NUMBER IN THE RANGE	
4565	BF3B	BE Ø1	16	LBF3B	LDX	RVSEED+1	* MOVE VARIABLE
4566		9F 50			STX	FPAØ	* RANDOM NUMBER
4567	BF4Ø	BE Ø1			LDX	RVSEED+3	* SEED TO
4568		9F 52			STX	FPAØ+2	* FPAØ
4569	BF45	BE BF	74	LBF45	LDX	RSEED	= MOVE FIXED
457Ø	BF48	9F 5D	)		STX	FPA1	= RANDOM NUMBER
4571	BF4A	BE BF	76		LDX	RSEED+2	= SEED TO
4572	BF4D	9F 5F	:		STX	FPA1+2	= MANTISSA OF FPAØ
4573	BF4F	BD BA	DØ		JSR	LBADØ	MULTIPLY FPAØ X FPA1
4574	BF52	DC AD	1		LDD	VAD	GET THE TWO LOWEST ORDER PRODUCT BYTES
4575	BF54	C3 65	8B		ADDD	#\$658B	ADD A CONSTANT
4576	BF57	FD Ø1			STD	RVSEED+3	SAVE NEW LOW ORDER VARIABLE RANDOM # SEED
4577	BF5A	DD 52			STD	FPAØ+2	SAVE NEW LOW ORDER BYTES OF FPAØ MANTISSA
4578	BF5C	DC AB			LDD	VAB	GET 2 MORE LOW ORDER PRODUCT BYTES
4579		C9 B0				#\$BØ	ADD A CONSTANT
4580	BF6Ø	89 Ø5			ADCA		ADD A CONSTANT
4581	BF62	FD Ø1			STD	RVSEED+1	SAVE NEW HIGH ORDER VARIABLE RANDOM # SEED
4582	BF65	DD 50			STD	FPAØ	SAVE NEW HIGH ORDER FPAØ MANTISSA
4583	BF67	ØF 54			CLR	FPØSGN	FORCE FPAØ MANTISSA = POSITIVE
4584	BF69	86 80			LDA STA	#\$8Ø	* SET FPAØ BIASED EXPONENT
4585 4586	BF6B BF6D	97 4F 96 15			LDA	FPØEXP FPA2+2	* TO Ø 1 < FPAØ < Ø GET A BYTE FROM FPA2 (MORE RANDOMNESS)
4587	BF6F	97 63			STA	FPSBYT	SAVE AS SUB BYTE
4588		7E BA			JMP	LBA1C	NORMALIZE FPAØ
4589	5171	, , ,	. 10	*	0111	EBATO	NOMINEIZE ITAD
4590	BF74	40 E6	i	RSEED	FDB	\$4ØE6	*CONSTANT RANDOM NUMBER GENERATOR SEED
4591		4D AB			FDB	\$4DAB	*
4592							
4593				* SIN			
4593 4594					N FUNCT	ION REQUIRES AN ARGUMENT I	N RADIANS AND WILL REPEAT ITSELF EVERY
				* THE SI			N RADIANS AND WILL REPEAT ITSELF EVERY Y 2*PI AND ONLY THE FRACTIONAL PART IS
4594 4595 4596				* THE SI * 2*PI R * RETAIN	ADIANS. ED. SIN	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE
4594 4595 4596 4597				* THE SI * 2*PI R * RETAIN	ADIANS. ED. SIN	THE ARGUMENT IS DIVIDED B	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE
4594 4595 4596 4597 4598				* THE SI * 2*PI R * RETAIN * MULTIP	ADIANS. ED. SIN LIED BY	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE 2*PI.
4594 4595 4596 4597 4598 4599				* THE SI * 2*PI R. * RETAIN * MULTIP * SIN IS	ADIANS. ED. SIN LIED BY EVALUA	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE 2*PI. C IDENTITIES BELOW:
4594 4595 4596 4597 4598 4599 4600				* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3*	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE 2*PI. C IDENTITIES BELOW: PI)/2+X)
4594 4595 4596 4597 4598 4599 4600 4601	BF78	BD BC		* THE SI * 2*PI R. * RETAIN * MULTIP * SIN IS	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI 	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE 2*PI. C IDENTITIES BELOW: PJ1/2+X) COPY FPAØ TO FPA1
4594 4595 4596 4597 4598 4599 4600 4601 4602	BF78 BF7B	8E BF	BD	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR LDX	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603	BF78 BF7B BF7E	8E BF D6 61	BD	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI JSR LDX LDB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP1SGN	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2+X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604	BF78 BF7B BF7E BF8Ø	8E BF D6 61 BD BB	BD 89	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI JSR LDX LDB JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF  TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBCSF #LBFBD FP15GN LBB89	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PTI)/2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605	BF78 BF7B BF7E BF8Ø BF83	8E BF D6 61 BD BB BD BC	BD 89 5F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI JSR LDX LDB JSR JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBCSF #LBFBD FP1SGN LBB89 LBCSF	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI)/2+X)  COPY FPAØ TO FPA1 POINT (X) TO 2*PI *6ET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4606	BF78 BF7B BF7E BF8Ø BF83 BF86	8E BF D6 61 BD BB BD BC 8D B0	BD	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI JSR LDX LDB JSR JSR BSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF  TED USING THE TRIGONOMETRI X.) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBB FP15GN LBB89 LBC5F LBF38	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *6ET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4606 4607	BF78 BF7B BF7E BF80 BF83 BF86 BF88	8E BF D6 61 BD BB BD BC 8D B0 ØF 62	89 5 5 F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI JSR LDX LDB JSR JSR BSR CLR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF  TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBF3D FP1SGN LBB89 LBC5F LBF38 RESSGN	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2+X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIOE FPAØ BY 2*PI COPY FPAØ TO FPA1 COPY FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE
4594 4595 4596 4597 4598 4600 4601 4602 4603 4604 4605 4605 4606 4607 4608	BF78 BF7B BF7E BF83 BF86 BF88 BF88	8E BF D6 61 BD BB BD BC 8D BØ ØF 62 96 50	89 5 5F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI JSR LDX LDB JSR JSR BSR CLR LDA	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP1SGN LBB89 LBC5F LBF38 RESSGN FP1EXP	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2+X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA1
4594 4595 4596 4597 4598 4600 4601 4602 4603 4604 4604 4605 4606 4607 4608 4609	BF78 BF7B BF7E BF80 BF83 BF86 BF88 BF8A BF8C	8E BF D6 61 BD BB BD BC 8D BC 9F 62 96 50 D6 4F	89 5 5F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X):	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR LDX LDB JSR JSR SSR CLR LDA LDB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF  TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBF3D FP1SGN LBB89 LBC5F LBF38 RESSGN	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA1 *GET EXPONENT OF FPA1 *GET EXPONENT OF FPA0
4594 4595 4596 4597 4598 4600 4601 4602 4603 4604 4605 4605 4606 4607 4608	BF78 BF7B BF7E BF80 BF83 BF86 BF88 BF8A BF8C	8E BF D6 61 BD BB BD BC 8D BØ ØF 62 96 50	89 5 5F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR LDX LDB JSR JSR BSR CLR LDA LDB JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP1SKN LBB89 LBC5F LBF38 RESSGN FP1EXP FP0EXP	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: P1)/2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO FPA1 I CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1
4594 4595 4596 4598 4599 4600 4601 4602 4603 4604 4605 4606 4606 4607 4608 4609 4610	BF78 BF7B BF7E BF83 BF83 BF86 BF88 BF88 BF8A BF8C BF8E	8E BF D6 61 BD BB BD BC 8D BC 9F 62 96 50 D6 4F	89 5 5 F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR LDX LDB JSR JSR BSR CLR LDA LDB JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF  TED USING THE TRIGONOMETRI	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: P1)/2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO FPA1 I CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1
4594 4595 4596 4597 4598 4699 4600 4601 4602 4603 4604 4605 4606 4607 4608 4609 4610 4611	BF78 BF7B BF7E BF80 BF83 BF86 BF88 BF8A BF8C BF8E	8E BF D6 61 BD BB BD BC 8D BØ ØF 62 96 50 D6 4F BD B9	89 5 5 F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PII JSR LDX LDB JSR JSR BSR CLR LDA LDB JSR LDA LDB JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP1SGN LBBB9 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2+X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SET RESULT SIGN = POFITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4606 4607 4608 4610 4611 4612 4613 4614	BF78 BF78 BF78 BF80 BF83 BF86 BF88 BF88 BF88 BF88 BF91 BF91 BF97	8E BF D6 61 BD BB BD BC 8D BØ ØF 62 96 50 D6 4F BD B9 8E BF BD B9 96 54	89 5 5 F 8 B C 8 C C 2 B 9	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LED BY EVALUA =SIN(PI JSR LDB JSR JSR BSR CLR LDA LDB JSR LDA LDS LDB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBF6D FP1SGN LBB69 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2+X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ
4594 4595 4596 4597 4598 4599 4601 4602 4603 4604 4605 4606 4607 4608 4609 4611 4612 4613 4614 4615	BF78 BF7B BF8Ø BF83 BF86 BF88 BF8A BF8C BF8E BF91 BF99	8E BF D6 61 BD BB BD BC 8D BØ ØF 62 96 50 D6 4F BD B9 8E BF BD B9 96 54 34 Ø2	89 5 F 6 BC 7 C2 8 B9	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI JSR LDX LDB JSR JSR BSR CLR LDA LDB JSR LDB JS	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP1SGN LBB89 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC LINS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI)/2*X)  COPY FPAØ TO FPA1 POINT (X) TO 2*PI *6ET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *6ET EXPONENT OF FPA0 *SET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK
4594 4595 4597 4598 4599 4600 4601 4602 4603 4604 4606 4607 4608 4610 4611 4612 4613 4614 4615 4615	BF78 BF78 BF7E BF80 BF83 BF86 BF88 BF80 BF81 BF94 BF94 BF99 BF99	8E BF D6 61 BD BB BD BC 8D B0 ØF 62 96 50 D6 4F BD B9 8E BF BD B9 96 54 34 Ø2 2A Ø9	89 5F BC C2 B9	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY  EVALUA =SIN(PI  JSR LDX LDB JSR SSR BSR CLR LDA LDB LDA LDB JSR LDB LDB JSR LDB LDB JSR LDB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP15KN LBB89 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA1 *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4607 4608 4607 4610 4611 4611 4611 4611 4615 4614 4615 4616 4617	BF78 BF78 BF80 BF86 BF86 BF86 BF86 BF87 BF97 BF99 BF99 BF99	8E BF D6 61 BD BB BD BC 8D BØ ØF 62 96 5C D6 4F BD B9 8E BF BD B9 96 54 34 Ø2 2A Ø9 BD B9	89 5 5 F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR LDX LDB JSR BSR CLR LDA LDB JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBF6D FP1SGN LBB89 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBF66 LB9B4	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PTI)/2*X)  COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4607 4608 4607 4611 4612 4611 4612 4613 4614 4615 4616 4617 4618	BF78 BF7B BF78 BF80 BF80 BF80 BF8A BF8C BF91 BF94 BF99 BF99 BF98 BF99 BF98	8E BF BD B9 BB BD B9 BB	BD 89 5F 6F	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR LDX LDB JSR JSR CLR LDA LDB JSR CH LDA LDB JSR CH LDA LDB JSR CH LDA LDB JSR LDA LDB LDA	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP1SGN LBB89 LBC5F LBF38 RESSGN FP1EXP FP0EXP LB9BC LNS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FP0SSN A LBFA6 LB9B4 FP0SGN	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2+X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 IRT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ GET SIGN OF FPAØ GET SIGN OF FPAØ GET SIGN OF FPAØ
4594 4595 4596 4597 4598 4599 4601 4602 4603 4606 4606 4608 4609 4610 4611 4612 4613 4614 4615 4616 4617 4616 4617	BF78 BF78 BF78 BF83 BF83 BF86 BF88 BF98 BF99 BF99 BF99 BF99 BF98 BF98	8E BF BD BC BC BD BC	89 5F 8C C2 89	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN	ADIANS. ED. SIN LIED BY EVALUA =SIN(PI JSR LDX LDB JSR BSR CLR LDA LDB JSR LDA LDS BPL JSR LDA LDS BBL JSR LDA BBL	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP15GN LBBB9 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LB9B4 FPØSGN LBFA9	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI//2*X' COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 *GET EXPONENT OF FPAØ *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ BRANCH IF NEGATIVE
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4607 4608 4607 4610 4611 4612 4613 4614 4615 4617 4616 4617 4618 4619 4619 4620	BF78 BF78 BF78 BF80 BF86 BF86 BF86 BF91 BF97 BF99 BF98 BF90 BF70 BF70 BF70 BF70 BF70 BF70 BF70 BF7	8E BF D6 61 BD B6 B0	89 5F BC C2 89	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN **NOW FPA	ADIANS. D. SIN BY EVALUA ESIN(PI JSR LDA LDB JSR BSR CLR LDA LDB JSR LDA LDD LDS LDA LDD LDS LDA LDD LDS LDA LDD LDA LDA LDB JSR LDA LDD LDA LDD LDA LDB LDB LDA LDB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP15GN LBB89 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LB9B4 FPØGGN LBFA9 RELFLG	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*P1, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PI)/2*X) C COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA1 *GET EXPONENT OF FPA1 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ GET SIGN OF FPAØ BRANCH IF NEGATIVE COM IF +(3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG)
4594 4595 4596 4597 4598 4599 4600 4601 4602 4604 4605 4604 4605 4607 4608 4610 4611 4612 4613 4614 4615 4616 4617 4618 4619 4620 4621	BF78 BF78 BF88 BF88 BF88 BF86 BF86 BF94 BF97 BF99 BF98 BF98 BF98 BF48 BFA6 BFA6	8E BF BD B9 BB	89 5F 8C C2 89	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN *NOW FPA	ADIANS. D. SIN LIED BY EVALUA ES JUSE LOX LOB LOB LOB LOB LOB BSR LOB LOB LOB JSR LOB LOB LOB JSR LOB LOB JSR LOB LOB JSR LOB BSR LOB LOB JSR LOB LOB JSR CCR LOB LOB JSR CCR LOB LOB JSR CCR LOB LOB JSR CCR LOB LOB JSR LOB LOB JSR LOB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP1SGN LBB89 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LB9B4 FPØSGN LBFA9 RELFLG LBEE9	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI)/2+X)  COPY FPAØ TO FPA1 POINT (X) TO 2*PI  *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI  COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ GET SIGN OF FPAØ BRANCH IF NEGATIVE COM IF *(3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG) TOGGLE MANTISSA SIGN OF FPAØ
4594 4595 4596 4597 4598 4599 4601 4602 4604 4605 4606 4606 4607 4608 4610 4611 4612 4613 4614 4615 4616 4616 4617 4618 4619 4620 4621 4622	BF78 BF78 BF78 BF83 BF86 BF88 BF86 BF86 BF99 BF99 BF99 BF98 BF99 BF98 BF40 BFA4 BFA4 BFA4 BFA4	8E BF D6 61 BD BB BD BC ØF 62 96 50 D6 4F BD B9 96 54 34 Ø2 BD B9 96 54 22A Ø9 BD B9 96 54 28 Ø5 Ø3 ØA BD BB BD BD BB BD BB BD BD BBB BB BB BB BB BB BB BB BB BB BB BB BB	BD 89 5F 8C C2 B9 84 84 84 85 85 85 85 85 85 85 85 85 85 85 85 85	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN **NOW FPA	ADIANS. D. SIN LIED BY EVALUA SIN LIED BY LOX LDB JSR LDX LDB JSR LDA LDB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBCSF #LBFBD FPISGN LBB89 LBC5F LBF38 RESSGN FPIEXP FPØEXP LB9BC LBSONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LB9B4 FPØSGN LBFA9 RELFLG LBFE9 #LBFC2	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI//2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *6ET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *6ET EXPONENT OF FPA0 *SET RESULT SIGN = POSITIVE *6ET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 IRT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ GET SIGN OF FPAØ BRANCH IF NEGATIVE COM IF +(3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG) POINT X TO FP (.25)
4594 4595 4597 4598 4599 4600 4601 4602 4603 4604 4606 4607 4608 4607 4610 4611 4611 4613 4614 4615 4614 4617 4618 4617 4618 4620 4620 4622 4622 4623	BF78 BF78 BF78 BF83 BF83 BF86 BF88 BF91 BF94 BF99 BF99 BF98 BF42 BFA4 BFA6 BFA6	8E BF D6 61 BD BB BD BC 96 5C D6 4F BD B9 8E BF BD B9 96 54 24 02 24 02 24 05 96 54 28 05 03 0A BD BE 8E BF BD B9 96 54 8E BF BD B9 96 54 8E BF BD B9 96 54 8E BF BD B9 8E BF BD B9 BD B0 BD B0 B0 BD B0 BD B0 B0 BD B0 B0 B0 B0 B0 B0 B0 B0 B0 B0 B0 B0 B0 B	89 5F 8C C2 89 6 C2 C2 C2 C2 C2	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN *NOW FPA	ADIANS. D. SIN BY EVALUA SIN IPED BY LDX LDB JSR LDX LDB JSR LDA LDS JSR LDA LDS JSR LDA LDS JSR LDA LDS JSR LDA JSR	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP15KN LBBB9 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC LBSONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LB9B4 FPØSGN LBFA9 RELFLG LBEE9 #LBFC2 LB9C2	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI)/2+X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA1 *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ GET SIGN OF FPAØ BRANCH IF NEGATIVE COM IF +(3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG) TOGGLE MANTISSA SIGN OF FPAØ POINT X TO FP (.25) ADD .25 (PI/2) TO FPAØ
4594 4595 4596 4597 4598 4599 4601 4602 4603 4604 4605 4606 4607 4608 4610 4611 4611 4612 4613 4614 4615 4616 4617 4618 4617 4618 4619 4620 4621 4622 4623 4623 4624	BF78 BF78 BF88 BF88 BF86 BF86 BF86 BF97 BF99 BF99 BF98 BF94 BFA2 BFA2 BFA2 BFA6 BFA6 BFA6 BFA6 BFA6 BFA6 BFA6 BFA6	8E BF BB B9 BB B9 BB B9 BB B9 BB BB BB BB BB	89 5F 8C C2 89 6 C2 C2 C2 C2 C2	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN *NOW FPA	ADIANS. D. SIN LIED BY EVALUAM SIN(PI JSR LDX LDB JSR LDX JSR LDA LDB LDB LDB LDB LDB JSR CCR LDA LDB LDB LDB LDB LDB LDB LDB JSR CCR LDA LDB	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP15KN LBBB9 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC LBSONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LB9B4 FPØSGN LBFA9 RELFLG LBEE9 #LBFC2 LB9C2	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: P!)/2+X)  COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA1 *GET EXPONENT OF FPA1 *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ GET SIGN OF FPAØ BRANCH IF NEGATIVE COM IF + (3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG) TOGGLE MANTISSA SIGN OF FPAØ POINT X TO FP (.25) ADD .25 (PI/2) TO FPAØ GET OLD MANTISSA SIGN
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4606 4607 4608 4609 4611 4612 4613 4614 4615 4614 4615 4616 4617 4618 4619 4620 4622 4622 4622 4622 4624 4625	BF78 BF78 BF78 BF80 BF80 BF80 BF80 BF80 BF94 BF99 BF99 BF99 BF40 BFA0 BFA0 BFA0 BFA0 BFA0 BFA0 BFA0	8E BF BD B9 BB	BD 89 5F 8C 8C 89 89 84 84 85 85 85 85 85 85 85 85 85 85 85 85 85	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN *NOW FPA	ADIANS. D. SIN LIED BY EVALUA ES JOSR LDX LDB JSR LDX LDB LDB BSR LDA LDB BSR LDA LDB BSR LDA LDB LDB LDB LDB LDB LDB LDB LDD JSR LDA LDB LDB LDB LDB LDD JSR LDS	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBCSF #LBFBD FPISGN LBB89 LBC5F LBF38 RESSGN FPIEXP FPØEXP LB9BC LNS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LBFA6 LBPB4 FPØSGN LBFA9 RELFLG LBEE9 #LBFC2 LB9C2 A	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI//2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ BRANCH IF NEGATIVE COM IF +(3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG) TOGGLE MANTISSA SIGN OF FPAØ POINT X TO FP (.25) ADD .25 (PI/2) TO FPAØ GET SIGN OF MANTISSA SIGN OF FPAØ POINT X TO FP (.25) ADD .25 (PI/2) TO FPAØ GET OLD MANTISSA SIGN * BRANCH IF OLD
4594 4595 4597 4598 4599 4600 4601 4602 4602 4603 4604 4606 4607 4608 4607 4610 4611 4611 4611 4615 4614 4615 4617 4618 4619 4620 4620 4622 4623 4624 4625 4625 4625	BF78 BF78 BF78 BF83 BF83 BF86 BF88 BF94 BF97 BF99 BF98 BF98 BF98 BFA4 BFA4 BFA4 BFA5 BFA6 BFA7 BFA6 BFA7 BFA6 BFA8	8E BF BD BB	BD 89 5F 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN *NOW FPA	ADIANS. D. SIN	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X.) & -SIN(PI/2-X)=SIN((3* LBC5F #LBFBD FP15GN LBBB9 LBC5F LBF38 RESSGN FP1EXP FPØEXP LB9BC INS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LB9B4 FPØSGN LBFA9 RELFLG LBEE9 #LBFC2 LB9C2 A LBFB7	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI/2/2X)  COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 *GET EXPONENT OF FPAØ *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ BERNCH IF NEGATIVE COM IF +(3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG) TOGGLE MANTISSA SIGN OF FPAØ POINT X TO FP (.25) ADD .25 (PI/2) TO FPAØ GET OLD MANTISSA SIGN **BRANCH IF OLD **SIGN WAS POSITIVE
4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4606 4607 4608 4609 4611 4612 4613 4614 4615 4614 4615 4616 4617 4618 4619 4620 4622 4622 4622 4622 4624 4625	BF78 BF78 BF78 BF80 BF80 BF80 BF80 BF94 BF97 BF99 BF98 BF90 BF40 BF42 BF42 BF42 BF42 BF46 BF46 BF46 BF47 BF48 BF48 BF48 BF48 BF48 BF48 BF48 BF48	8E BF BD B9 BB	89 5F 6 8C 6 C2 89 89 6 E9 6 C2 6 C	* THE SI * 2*PI R * RETAIN * MULTIP * SIN IS * SIN(X): SIN *NOW FPA	ADIANS. D. SIN LIED BY EVALUA ES JOSR LDX LDB JSR LDX LDB LDB BSR LDA LDB BSR LDA LDB BSR LDA LDB LDB LDB LDB LDB LDB LDB LDD JSR LDA LDB LDB LDB LDB LDD JSR LDS	THE ARGUMENT IS DIVIDED B CE THE ARGUMENT WAS DIVIDE THE APPROPRIATE POWER OF TED USING THE TRIGONOMETRI -X) & -SIN(PI/2-X)=SIN((3* LBCSF #LBFBD FPISGN LBB89 LBC5F LBF38 RESSGN FPIEXP FPØEXP LB9BC LNS ONLY THE FRACTIONAL PA #LBFC2 LB9B9 FPØSGN A LBFA6 LBFA6 LBPB4 FPØSGN LBFA9 RELFLG LBEE9 #LBFC2 LB9C2 A	Y 2*PI AND ONLY THE FRACTIONAL PART IS D BY 2*PI, THE COEFFICIENTS MUST BE 2*PI.  C IDENTITIES BELOW: PPI//2*X) COPY FPAØ TO FPA1 POINT (X) TO 2*PI *GET MANTISSA SIGN OF FPA1 *AND DIVIDE FPAØ BY 2*PI COPY FPAØ TO FPA1 CONVERT FPAØ TO AN INTEGER SET RESULT SIGN = POSITIVE *GET EXPONENT OF FPA0 *SUBTRACT FPAØ FROM FPA1 RT OF ARGUMENT/2*PI POINT X TO FP (.25) SUBTRACT FPAØ FROM .25 (PI/2) GET MANTISSA SIGN OF FPAØ SAVE IT ON STACK BRANCH IF MANTISSA POSITIVE ADD .5 (PI) TO FPAØ BRANCH IF NEGATIVE COM IF +(3*PI)/2 >= ARGUMENT >+ PI/2 (QUADRANT FLAG) TOGGLE MANTISSA SIGN OF FPAØ POINT X TO FP (.25) ADD .25 (PI/2) TO FPAØ GET SIGN OF MANTISSA SIGN OF FPAØ POINT X TO FP (.25) ADD .25 (PI/2) TO FPAØ GET OLD MANTISSA SIGN * BRANCH IF OLD

4629	BFBA	7 E	ΒE	FØ					JMP	LBEFØ	GO CALCULATE POLYNOMIAL VALUE
4630											
4631	BFBD	83	49	ØF	DA	A2		LBFBD	FCB	\$83,\$49,\$ØF,\$DA,\$A2	6.28318531 (2*PI)
4632	BFC2	7 F	ØØ	ØØ	ØØ	ØØ		LBFC2	FCB	\$7F,\$00,\$00,\$00,\$00	.25
4633											
4634								* MODIFIE	D TAYLO	OR SERIES SIN COEFFICIENTS	
4635	BFC7	Ø5						LBFC7	FCB	6-1	SIX COEFFICIENTS
4636	BFC8	84	E6	1A	2 D	1B		LBFC8	FCB	\$84,\$E6,\$1A,\$2D,\$1B	* -((2*PI)**11)/11!
4637	BFCD	86	28	Ø7	FB	F8		LBFC8	FCB	\$86,\$28,\$Ø7,\$FB,\$F8	* ((2*PI)**9)/9!
4638	BFD2	87	99	68	89	Ø1		LBFD2	FCB	\$87,\$99,\$68,\$89,\$01	* -((2*PI)**7)/7!
4639	BFD7	87	23	35	DF	Ε1		LBFD7	FCB	\$87,\$23,\$35,\$DF,\$E1	* ((2*PI)**5)/5!
4640	BFDC	86	Α5	5 D	E7	28		LBFDC	FCB	\$86,\$A5,\$5D,\$E7,\$28	* -((2*PI)**3)/3!
4641	BFE1	83	49	ØF	DA	A2		LBFE1	FCB	\$83,\$49,\$ØF,\$DA,\$A2	* 2*PI
4642											
4643	BFE6	Α1	54	46	8F	13	8F	LBFE6	FCB	\$A1,\$54,\$46,\$8F,\$13	UNUSED GARBAGE BYTES
4644	BFEC	52	43	89	CD				FCB	\$8F,\$52,\$43,\$89,\$CD	UNUSED GARBAGE BYTES
4645											
4646								* INTERRU	PT VEC	TORS	
4647	BFFØ	Α6	81					LBFFØ	FDB	LA681	RESERVED
4648	BFF2	Ø1	ØØ					LBFF2	FDB	SW3VEC	SWI3
4649	BFF4	Ø1	Ø3					LBFF4	FDB	SW2VEC	SWI2
4650	BFF6	Ø1	ØF					LBFF6	FDB	FRQVEC	FIRQ
4651	BFF8	Ø1	ØC					LBFF8	FDB	IRQVEC	IRQ
4652	BFFA	Ø1	Ø6					LBFFA	FDB	SWIVEC	SWI
4653	BFFC	Ø1	Ø9					LBFFC	FDB	NMIVEC	NMI
4654	BFFE	ΑØ	27					LBFFE	FDB	RESVEC	RESET

MODIFIED REGISTERS	ADDRESS	DESCRIPTION
ALL	RESVEC (AØ27)	RESET ENTRY POINT - set SAM, PIAs ram check - if RSTFLG = $\$55$ , then jump to RSTVEC; otherwise do a cold start.
ALL	BACDST (AØ74)	COLD START ENTRY POINT - clear 1st 1K of RAM, reset BASIC's pointers.
ALL	BAWMST (AØE8)	WARM START ENTRY POINT - reset some of BASIC's pointers.
Α	A171	ASCII CONSOLE IN - read a character from CONSOLE IN. Mask off bit 7 and return character in ACCA.
Α	A176	CONSOLE IN - read a character from active input device (DEVNUM). Return character in ACCA.
A,B,X	A199	CURSOR DRIVER - put a cursor block on the screen at the address in CURPOS.
Α	A1B1	WAIT FOR A KEYSTROKE - blink the cursor while waiting for a keystroke. Return the ASCII value of the key in ACCA when a key is depressed.
Α	KEYIN (A1CB)	SCAN THE KEYBOARD FOR A KEY DEPRESSION - Return zero flag = 1 if no new key down. Return the ASCII value of the key in ACCA if a new key is depressed.
NONE	PUTCHR (A282)	CONSOLE OUT - sends character in ACCA to output device. The output device is specified in DEVNUM.
NONE	A2BF	RS232 OUTPUT DRIVER - software UART specifically formatted to drive a line printer. The routine may be used to drive other devices, and has been modified by all revisions to BASIC. Version 1.2 will not begin transmitting data until the destination device is ready.
NONE	A3ØA	PUT A CHARACTER ON THE SCREEN - place a character on the screen (the screen starts at \$400) at the location stored in CURPOS.
NONE	A35F	INITIALIZE PRINT PARAMETERS - set up tab field width, tab zone, current position and line width according to the device selected (DEVNUM). This routine will vector into RAM at RVEC2.
A,B,X	A38D	GET A BASIC INPUT LINE - this routine will allow the

COLOR BASIC UNRAVELLED II		APPENDIX C ORIGIN:SPECTRAL ASSOC BASIC ROUTINES AND ENTRY POINTS REVISED:12/13/1999 WALTER K ZYDHEK
		inputting of a BASIC input line from CONSOLE IN.
Α	A3ED	INPUT DEVICE NUMBER CHECK - check for a valid input device number and file mode. This routine will vector into RAM at RVEC5.
Α	A406	OUTPUT DEVICE NUMBER CHECK - check for a valid output device number and file mode. This routine will vector into RAM at RVEC6.
Α	A42D	CLOSE A FILE - closes the file specified by DEVNUM. Vectors into RAM at RVEC8.
Α	A549	BREAK CHECK - check to see if the break key or the pause (shift $@$ ) key is down. Vectors into RAM at RVEC11.
А,В,Х	A7Ø1	START TAPE, READ A BLOCK - load a block of data from the cassette tape into RAM and return error status in ACCB.
А,В,Х	GETBLK (A7ØB)	READ A BLOCK - load a block of data from the cassette tape into RAM.
А,В,Х	CASON (A77C)	SYNC THE TAPE DECK - turn on the tape and wait for sync bytes.
X	A7D1	LONG DELAY – approximately $1/2$ second delay. Loads X register with zero and counts it down to zero.
A,B,X,Y	WRLDR (A7D8)	WRITE LEADER - write a leader to the cassette tape.
A,B,X,Y	SNDBLK (A7F4)	WRITE BLOCK TO CASSETTE - take a block of RAM and write it to cassette.
A,B,X,Y	A82A	WRITE A BYTE TO TAPE - write ACCA to tape. This routine does the "dirty work" of actually writing a byte to tape.
В,Х	A928	CLEAR SCREEN - store blanks ( $\$60$ ) to the video display screen.
Α	A974	DISABLE ANALOG MULTIPLEXER - disable the sound analog multiplexer. This will not allow any sound input to pass through to the analog multiplexer.
Α	A976	ENABLE ANALOG MULTIPLEXER - enable the sound analog multiplexer. This will allow sound inputs to pass through the analog multiplexer.

COLOR BASIC UNRAVELLED	) II	APPENDIX C ORIGIN:SPECTRAL ASSOC BASIC ROUTINES AND ENTRY POINTS REVISED:12/13/1999 WALTER K ZYDHEK
A,U	A9A2	SET ANALOG MULTIPLEXER - set the control inputs to the analog multiplexer to allow one of the four inputs to pass through.
A,B,X,U	GETJOY (A9DE)	READ JOYSTICKS - software 6-bit analog to digital conversion routine used to read the joystick potentiometers.
A,B	AC33	FREE RAM CHECK - check to see if there is room to store 2*ACCB bytes in free RAM, OM error if not.
A,B,X,U	ADØ1	LINE NUMBER SEARCH - search the BASIC program for the line number stored in BINVAL. Set the carry flag if no match.
X	AD26	ERASE VARIABLES - erase BASIC's variables and reset pointers.
А,В,Х	AF67	CONVERT LINE NUMBER TO BINARY - convert an ASCII line number in a BASIC program to binary and return the value in BINVAL.
A,B,X,Y	AFA4	PUT STRING IN STRING SPACE - move a string whose descriptor is located at FPAØ+2 into the string space.
NONE	B143	NUMERIC TYPE MODE CHECK - test the contents of VALTYP and return if positive. TM error if negative.
NONE	B146	STRING TYPE MODE CHECK - test the contents of VALTYP and return if negative. TM error if positive.
ALL	B156	${\sf EVALUATE}$ ${\sf EXPRESSION}$ - evaluate an expression in a BASIC statement. BASIC's input pointer must be pointed to the expression.
ALL	B357	EVALUATE VARIABLE - evaluate the variable to which BASIC's input pointer is pointing. Return with X and VARPTR pointing to the variable descriptor. If the variable is not stored in the variable table, that variable name with a value of zero is inserted into the variable table.
Α	B3A2	SET CARRY IF NOT ALPHA - set the carry flag if ACCA is not and ASCII alpha character.
А,В,Х	INTCNV (B3ED)	CONVERT FPAØ TO INTEGER. Convert FPAØ to a signed 2-byte integer; return the value in ACCD.
А,В,Х	GIVABF (B4F4)	CONVERT INTEGER TO FLOATING POINT - convert the value in ACCD into a floating point number in FPAØ.

COLOR BASIC UNRAVEL	LED II	APPENDIX C ORIGIN:SPECTRAL ASSOC BASIC ROUTINES AND ENTRY POINTS REVISED:12/13/1999 WALTER K ZYDHEK
А,В,Х	B54C	PUT DESCRIPTOR ON STRING STACK - put the direct page descriptor buffer data (STRDES) onto the string stack. Set the variable type (VALTYP) to string type.
А,В,Х	B56D	RESERVE ACCB BYTES IN STRING SPACE - reserve ACCB bytes in the string storage space. Return with the starting address of the reserved string space in X and FRESPC.
А,В,Х	B74Ø	INTEGER SIZE CHECK - check FPAØ to make sure it is in the range $-32768 \ll$ FPAØ $\ll$ 32767. If it is, return the value of that integer in X.
ALL	B7C2	UNCRUNCH - uncrunch a basic line into BASIC's line input buffer. Vectors into RAM at RVEC24.
ALL	B821	CRUNCH - crunch the line that the input pointer is pointing to into the line input buffer and return the length of the crunched line in ACCD.
A,B,X,U	B99C	SEND STRING TO CONSOLE OUT - parse a string which is pointed to by ${\sf X}$ and send it to CONSOLE OUT.
А,В,Х	BD12	CONVERT STRING TO FLOATING POINT - convert an ASCII string pointed to by BASIC's input pointer into a floating point value in FPAØ. Vectors into RAM at RVEC19.
A,B,X,U	BDCC	PRINT DECIMAL NUMBER TO CONSOLE OUT - convert the value in ACCD into a decimal number and send it to CONSOLE OUT.
A,B,X,U	BDD9	FLOATING POINT TO ASCII STRING - convert the floating point number in FPA0 into an ASCII string in the string buffer.
A,B,X	BEFF	EXPAND THE POLYNOMIAL - calculate the value of an expanded polynomial expression. Enter with X pointing to a table of coefficients the first byte of which is the number of (coefficients -1) followed by that number of packed floating point numbers. The polynomial is evaluated as follows: value = $((FPAØ*YØ+Y1)*FPAØ+Y2)*FPAØ+YN)$

ADDRESS	DESCRIPTION
B9B4	ADD .5 TO FPAØ
B9B9	SUBTRACT FPAØ FROM FLOATING POINT NUMBER POINTED TO BY X, LEAVE RESULT IN FPAØ
В9ВС	ARITHMETIC OPERATION (-) JUMPS HERE - SUBTRACT FPAØ FROM FPA1 (ENTER WITH EXPONENT OF FPAØ IN ACCB)
B9C2	ADD FLOATING POINT NUMBER POINTED TO BY X TO FPAØ - LEAVE RESULTS IN FPAØ
B9C5	ARITHMETIC OPERATION (+) JUMPS HERE - ADD FPAØ TO FPA1 (ENTER WITH EXPONENT OF FPAØ IN ACCB AND EXPONENT OF FPA1 IN ACCA)
BA1C	NORMALIZE FPAØ
BA79	NEGATE FPAØ MANTISSA
BA83	ADD ONE TO FPAØ MANTISSA
BACA	ARITHMETIC OPERATION (*) JUMPS HERE - MULTIPLY FPAØ BY X - RETURN PRODUCT IN FPAØ
BADØ	MULTIPLY FPAØ MANTISSA BY FPA1, NORMALIZE HIGH ORDER BYTES OF PRODUCT IN FPAØ. THE LOW ORDER FOUR BYTES OF THE PRODUCT WILL BE STORED IN VAB-VAE
BB2F	UNPACK A FLOATING POINT NUMBER FROM X INTO FPA1
BB6A	FAST MULTIPLY FPAØ BY 10 AND LEAVE RESULT IN FPAØ
BB82	DIVIDE FPAØ BY 1Ø
BB8F	DIVIDE X BY FPAØ - LEAVE NORMALIZED QUOTIENT IN FPAØ
BB91	DIVIDE FPA1 BY FPAØ. ENTER WITH EXPONENT OF FPA1 IN ACCA AND FLAGS SET BY TSTA
BBA4	COMPARE FPAØ MANTISSA TO FPA1 MANTISSA - SET CARRY FLAG IF FPA1 >= FPAØ
ВСØВ	COPY MANTISSA FROM FPA2 TO FPAØ
BC14	COPY A PACKED FLOATING POINT NUMBER FROM X TO FPAØ
BC2A	PACK FPAØ AND SAVE IT IN FPA4

APPENDIX D	ORIGIN:SPECTRAL ASSOC
FLOATING POINT ROUTINES	REVISED:12/13/1999 WALTER K ZYDHEK

COLOR	RASTO	UNRAVELLED	ΤT
COLOR	DASIC	UNIKAVELLED	1 1

BC2F	PACK FPAØ AND SAVE IT IN FPA3
BC33	PACK FPAØ AND SAVE IT IN ADDRESS STORED IN VARDES
BC35	PACK FPAØ AND SAVE IT IN ADDRESS POINTED TO BY X
BC4A	MOVE FPA1 TO FPAØ RETURN WITH MANTISSA SIGN IN ACCA
BC5F	TRANSFER FPAØ TO FPA1
BC6D	CHECK FPA0; RETURN ACCB = 0 IF FPA0 = 0, ACCB = \$FF IF FPA0 = NEGATIVE, ACCB = 1 IF FPA0 = POSITIVE
BC7C	CONVERT A SIGNED NUMBER IN ACCB INTO A FLOTING POINT NUMBER
BC96	COMPARE A PACKED FLOATING POINT NUMBER POINTED TO BY X TO AN UNPACKED FLOATING POINT NUMBER IN FPAØ. RETURN ZERO FLAG SET AND ACCB=Ø IF EQUAL; ACCB = 1 IF FPAØ > (X); ACCB = \$FF IF FPAØ < (X)
BDØ9	FILL MANTISSA OF FPAØ WITH CONTENTS OF ACCB

START	END	DESCRIPTION
AØØØ	AØØD	INDIRECT JUMP TABLE
A1ØD	A128	DIRECT PAGE ROM IMAGE
A129	A146	PAGE ONE ROM IMAGE
A147	A17Ø	COPYRIGHT MESSAGES
A26E	A281	SPECIAL KEY LOOKUP TABLE
A85C	A87F	SINE WAVE LOOKUP TABLE
AA29	AA5Ø	SECONDARY DISPATCH TABLE
AA51	AA65	OPERATOR PRECEDENCE TABLE
AA66	AB19	PRIMARY RESERVED WORD TABLE
AB1A	AB66	SECONDARY RESERVED WORD TABLE
AB67	ABAE	PRIMARY DISPATCH TABLE
ABAF	ABEØ	ERROR MESSAGES
ABE1	ABEC	"ERROR IN" MESSAGE
ABED	ABF1	"OK" MESSAGE
ABF2	ABF8	"BREAK" MESSAGE
AFCF	AFD5	"?REDO" MESSAGE
BØE8	BØF7	"?EXTRA IGNORED" MESSAGE
B3DF	B3E3	FLOATING POINT NUMBER -32768
BAC5	BAC9	FLOATING POINT NUMBER 1.0
BB7D	BB81	FLOATING POINT NUMBER 10
BDB6	BDBA	FLOATING POINT NUMBER 999999999999999999999999999999999999
BDBB	BDBF	FLOATING POINT NUMBER 999999999
BDCØ	BDC4	FLOATING POINT NUMBER 1E+9
BECØ	BEC4	FLOATING POINT NUMBER .5

APPENDIX E	ORIGIN:SPECTRAL AS	SSOC
COLOR BASIC'S DATA/ASCII TABLE	ES REVISED:12/13/1999 WALTER K ZYD	HEK

BEC5	BEE8	TABLE OF MANTISSAS OF UNNORMALIZED POWERS OF TEN
BF74	BF77	CONSTANT RANDOM NUMBER SEED
BFBD	BFC1	FLOATING POINT NUMBER 2*PI
BFC2	BFC6	FLOATING POINT NUMBER .25
BFC7	BFE5	MODIFIED TAYLOR SERIES SINE COEFFICIENTS
BFE6	BFEF	TWO GARBAGE FLOATING POINT NUMBERS
BFFØ	BFFF	INTERRUPT VECTORS

COLOR BASIC UNRAVELLED II

### MEMORY MAP

One of the most important tools to have at your fingertips, if you are going to attempt to use any machine's built-in operating system, is a complete and accurate map of that system's memory structure. At the beginning of the BASIC disassembly listing you will find the most complete memory map available for the Color Computer outside of Microsoft's domain. It explains all of the variables in the direct page and what their functions are, defines all of the variables and buffers between the direct page and the video display RAM, and all of the variables that are used by Disk RAM. It identifies the areas in memory used by the variable tables, the array tables, the string space, cleared memory, and other important areas.

The direct page provides the most useful source of rapidly accessed variable space available in the 6809. When you become familiar with 6809 Machine Language programming you will notice that it is quicker byte wise and time wise to access variables which are located in the direct page. The Color Computer, of course, keeps the direct page in page 0, which makes it relatively compatible with 6502 programming. The direct page is also at the very bottom of RAM, where it is conveniently out of the way of any programs written by the user for whatever purpose he has in mind. If you look at the memory map at the beginning of Color BASIC Unravelled you will find that some of the variables will have asterisks designation in front of them. If that designation is PV, it defines a Permanent Variable. Permanent Variable has been chosen for lack of a better word; exactly what this means is that the variable has a defined function that is used by every command that BASIC has. Such a variable would be the beginning of BASIC, the top of free RAM, the beginning of array variables, the beginning of normal variables, the top of the string stock, the present pointer to the current value in the string stock and so forth. These variables will cause permanent harm to a BASIC program if they are modified during the course of your machine language program; therefore, you will have to be very careful how you use these variables. If you change the value, for instance, of the start of BASIC, you will cause BASIC to feel that there are fewer or more lines in the program than there actually are, which could easily result in an error or crash. Therefore, when you are making a program, which is designed to run and mesh with BASIC, you should not make any changes to a PV type variable, unless you are absolutely sure that you know what you are doing. Obviously, there can be some instances when you will want to change the start of BASIC. If there is some value you might want to change, you will have to make your decision based on what you're doing as to whether you want to change the variable or not, just be aware that PV variables are very tricky to change and changing them may blow up BASIC.

Other variables are designated TV, which is a temporary variable, and is a variable whose function should be uniform for all BASIC commands. A temporary variable has one specific use and one specific BASIC command. A perfect example would be the variable labeled DIMFLG This variable is used when defining a dimensioned array in order to specifically tell a certain routine in BASIC that the variable currently being defined is an array variable as opposed to a string variable or single precision variable. Once the variable is defined, obviously the value being stored in DIMFLG has no appropriate use for any other BASIC command. Therefore it may be modified if required for your own use, but you should be aware that the DIMFLG variable can be changed during the course of a BASIC program and that if you use it for some specific value in your program you may not harm BASIC, but don't expect the value to be unmodified by BASIC in the course of the normal operation of BASIC.

The variables not labeled either TV nor PV will be used by many different routines in BASIC and are neither Temporary nor Permanent, because they are used by so many routines that it doesn't matter what happens to these variables. These variables have a particular function, such as a pointer, an address counter or normal counter. They have been given a specific label because they do have a particular function that remains common from BASIC command to BASIC command. There are also variables, which are referred to as Scratch Pad variables. These variables have the designation VXX, where the XX is the actual Hex address of that variable in RAM. These are different from the temporary variables, in that they may be used for any particular function by any particular BASIC command; therefore they could be pointers, counters or addresses or any other kind of temporary storage. These are the most useful to use as temporary storage for your own routines, since if you modify these routines between various BASIC commands (not in a BASIC command but between BASIC commands) it will not cause any harm whatsoever to the operation of a BASIC program.

Other variables may be identified with a DV designation, which may be in conjunction with a TV or PV or may be by itself. This particular designation is used to define a double variable; that is, there are places in BASIC where a variable is loaded into a 16-byte register even though the variable is an 8-byte quantity. As such, the variable and the variable immediately following the DV designation may not be separated from each other; they must be immediately adjacent to one another in the memory space of the computer. If for any reason these variables are separated from one another in the memory space of the computer, the instructions in BASIC, which grab data from the double variable, will not function properly. There are some variables in Extended BASIC, which are two 16-byte quantities, which must be kept next to each other, such as HORBEG or VERBEG. This is necessary because an index register is pointed at the first of these variables, then it is incremented to the appropriate variable.

## AREA BETWEEN THE DIRECT PAGE AND VIDEO DISPLAY

The area between the direct page and the video display at \$400 is used by several different routines to store pieces of useful information. There are some large buffers and some small 2 or 3 byte value storage blocks, which are used by some specific routines and some that are in general used by many different routines. The interrupt jump vectors are stored from address \$100 to \$111. These contain the addresses where the interrupt factors jump to IRQ, FIRQ, NMI, RESET, and Software Interrupt routines. Immediately following the interrupt vectors are several small variables, which are used by different routines. The first of these is the USRJMP variable. These are three bytes, which Color BASIC uses to store the jump address for the USR function. When Extended BASIC is in the machine, these locations are not used by the USR function any more. They are instead used to store the timer value (TIMVAL). Timer value is only two bytes and Extended or Disk BASIC does not use the third byte of this three-byte block of data. The next variable stored is RVSEED, a five-byte value for the variable Random Number Seed. Following that is CASFLG, the Case Flag, which determines whether the characters being put on the screen are in upper or lower case. If the value stored here is zero you are in lower case. If it is \$FF they are in upper case. Then comes DEBVAL, which is the keyboard debounce delay value, a two-byte quantity. Following that is EXPJMP, which is the address that the EXEC command uses to jump to. Next are the command interpretation tables. There are normally only as many interpretation tables as there are ROMs plugged into the computer. If extended BASIC and not disk BASIC is plugged in, in the area following COMVEC are the USR function jump vectors for Extended BASIC. If Disk BASIC is plugged in, the USR jump vectors are transferred to the disk RAM. After the command interpretation tables is the keyboard buffer. This is the memory that is used so that there can be rollover in the keyboard

routine. Eight bytes are used to store the information on which keys have and have not been pressed. Following that are four bytes (POTVAL) to store the values of the joystick potentiometers. After POTVAL come BASIC's RAM vectors. An explanation of these vectors is provided in the memory map at the beginning of this book and the user should refer to that explanation in order to get a detailed description of how the RAM vectors function. Following the RAM hooks is a 40-byte block of data used to store string descriptors. This is the string stack, which is used in string manipulation functions. After the string stack comes the cassette file name buffer where the cassette file name is stored prior to searching for a cassette file. After that is a 256 byte block of data which is the cassette I/O buffer. Following that is a two-byte sub-block to the line input buffer, which is called the line input header. This is used to store the jump address of the next BASIC line. After line header comes LINBUF which is a 251-byte buffer to store BASIC input line as it is being typed in. This 251-byte area is also used for several different functions but primarily it is used as a line input buffer. The last block of data is a 41byte block of data following LINBUF up to the video display RAM which is called STRBUF. This is a string buffer, which is used to hold temporary string information and temporary strings before they are moved into the string space. It is most commonly used in floating point to ASCII string and ASCII string to floating point data conversions. Then from \$400 to \$5FF is the 512-byte block of video display RAM. If you have a disk hooked up to your system, the area from \$600 to \$989 is used by the disk for its own special I/O buffers and disk variables.

### INTERRUPTS

BASIC uses the 6809 interrupt structure to control those commands, which require precise timing intervals. The manner in which the interrupt signals are handled by the 6809 and 6821 (or 6822) in the Color Computer is described in the FACTS book and will not be covered in this book. Only the software aspects will be covered.

The 6821 Peripheral Interface, Adapter (PIA) may be programmed to pass either a 16.67 ms (60HZ) or a 63.5 microsecond input to the 6809's IRQ interrupt pin. Color BASIC uses only the 60HZ interrupt; the 63.5 microsecond input is never enabled. The IRQ routine is used to increment or decrement the following parameters:

COMMAND	VARIABLE	IRQ FUNCTION
SOUND	SNDDUR	Decrement SNDDUR
PLAY	PLYTMR	Decrement VD5 from PLYTMR
TIMER	TIMVAL	Increment TIMVAL
(DOS BASIC)	RDYTMR	Decrement RDYTMR

The SOUND and PLAY commands will fall into an endless timing loop which will only be terminated by the IRQ routine's decrementing SNDDUR and/or PLYTMR to zero.

FIRQ	The PIAs may be programmed to pass either the RS 232 status input or the cartridge interrupt
	signal to the 6809's FIRQ interrupt pin. Color
	BASIC uses FIRQ to vector control to a ROM-PAK if
	pins 7 & 8 of the cartridge port are connected
	together by a cartridge.
NMI	The 6809's NMI pin is connected only to the
	cartridge port. The Disk Operating System (DOS)
	uses the NMI to vector out of data transfers
	between the 6809 and the 1793 Floppy Disk
	Controller.
SWI	Not used by Color BASIC
SWI2	Not used by Color BASIC
SWI3	The DOS command of Disk BASIC calls SWI3. The
	user must provide a SWI3 servicing routine - Disk
	BASIC does not provide one.

# EXPRESSIONS AND OPERATORS

Rela	tional Operators	Arith	nmetic Operators
=	Equal	+	Add
<	Less Than	_	Subtract
>	Greater Than	*	Multiply
<=	Less Than or Equal	/	Divide
>=	Greater Than or Equal	^	Exponentiation
<>	Not Equal	-	Negation
Bool	ean Operators	Strin	ng Operators
AND		+	Concatenation
OR			
NOT			

# Rules for Evaluating Expressions

- 1. Operations of higher precedence are performed before operations of lower precedence. This means the multiplications and divisions are performed before additions and subtractions. As an example, 2+10/5 equals 4, not 2.4. When operations of equal precedence are found in a formula, the left-hand one is executed first: 6-3+5 = 8, not -2.
- 2. The order in which operations are performed can always be specified explicitly through the use of parentheses. For instance, to add 5 to 3 and then divide that by 4, we would use (5+3)/4, which equals 2. If, instead, we had used 5+3/4, we would get 5.75 as a result (5 plus 3/4).

The precedence of operators used in evaluating expressions is as follows, in order beginning with the highest precedence: (Note: Operators listed on the same line have the same precedence).

1	) FORMULAS ENCLOSED IN PARENTH	ESIS ARE ALWAYS EVALUATED FIRST.
2	<b>^</b>	EXPONENTATION
3	) NEGATION	-X WHERE X MAY BE A FORMULA
4	) */	MULTIPLICATION AND DIVISION
5	) +-	ADDITION AND SUBTRACTION
6	) RELATIONAL OPERATORS	= EQUAL
		<> NOT EQUAL
	(EQUAL PRECEDENCE FOR	< LESS THAN
	ALL SIX).	> GREATER THAN
		<= LESS THAN OR EQUAL
		>= GREATER THAN OR EQUAL
7	) NOT	LOGICAL AND BITWISE "NOT" LIKE
		NEGATION, NOT TAKES ONLY THE
		FORMULA TO ITS RIGHT AS AN
		ARGUMENT
8	) AND	LOGICAL AND BITWISE "AND"
9	) OR	LOGICAL AND BITWISE "OR"

The ASCII table is defined in Appendix J. It contains the order in which characters within the Color Computer are represented when two strings are compared. Characters within a set of strings are compared starting at the leftmost character to the end of the field specified.

Using the ASCII table, we can compare a string containing an "A" to one containing a "B" in the same position. The result is that the second string is greater than the first.

A string containing a blank is less than a "1", which is less than an "A", which is less than a "B". The string "A" is less than the string "ABC" or any string containing "A" as the first character. All characters are compared in sequence with the first unequal character defining the relationship between the strings. Thus, the same relational functions may be used for both strings and numbers.

Listed below are the differences between Color BASIC Version 1.0 and Version 1.2  $\,$ 

CHANGE	ADDRESS	Version 1.0	Version 1.2
A	\$AØØ1	\$C1	\$CB
В	\$AØ1B-\$AØC7	SEE LISTING 1	
С	\$A102-\$A104	LDU #\$A1Ø8	LEAY \$A108,PCR
D	\$A114	\$57	\$58
E	\$A155	\$3Ø	\$32
F	\$A15E	\$30	\$32
G	\$A1B5-\$A26D	SEE LISTING 2	
Н	\$A2C3-\$A2FA	SEE LISTING 3	
I	\$A44Ø	\$Ø8	\$Ø3
J	\$A56A	\$C1	\$CB
K	\$A6EB	\$07	\$14
L	\$ADFD	\$C1	\$CB
М	\$B23F	\$E8	\$E3
N	\$B38E	\$72	\$75
Р	\$B3ED-\$B427	SEE LISTING 4	
Q	\$B9D6	\$2B BMI	\$25 BCS

Change A is a branch length change caused by the keyboard driver mod.

Change B (Listing 1) is a major rework of the warm and cold start initializations required to allow the computer to accept 64K dynamic RAMs.

Change C was required because change B changed the storage location of the RESET jump vector from the U register to the Y register.

Change D modified the line printer baud rate.

Change E is the version number.

Change F is the copyright year.

Change G is a major change (Listing 2) to the keyboard driver allowing a quick scan of the keyboard if no keys are down and filtering of joystick button depressions out of the keyboard scan.

Change H is also a major change (Listing 3) which allows the line printer driver to output an eight bit character and will not allow any transmission until the receiving device is ready.

Change I causes an end of program block to be written to a cassette file if the buffer is empty when the file is closed.

Change J speeds up the INKEY\$ command by not entering the keyboard scan routine if no keys are depressed.

Change K is a minor change which causes the upper left hand corner of the screen to blink during cassette loading operations.

Change L speeds up the BREAK key check routine by not entering the

keyboard scan routine if no more keys are depressed.

Change M is a minor change which slightly speeds up the expression evaluation routine.

Change N is merely a different length branch caused by change P.

Change P (Listing 4) causes a numeric variable type check to be done before the INTCNV (FPA $\emptyset$  to integer) routine.

Change Q fixed a minor bug in the floating point addition routine.

*** LISTING 1				
AØ1B		BNE	BACDST	NO - DO A COLD START
AØ1D		LDX	RSTVEC	WARM START VECTOR
AØ1F		LDA	, X	GET FIRST BYTE OF WARM START ADDR
AØ21			,^ #\$12	IS IT A NOP?
AØ23		BNE	BACDST	NO - DO A COLD START
AØ25		JMP	X	YES, GO THERE
AWZS	*	UNF	, ^	1E3, GO THERE
AØ27	RESVEC	LDU	#LAØØE	BASIC WARM START ENTRY (RESET)
AØ2A	LAØ2A	CLRB		*
AØ2B		TFR	B,DP	* USE PAGE Ø AS DIRECT PAGE
	*****	****	*****	
AØ2D		LDX		POINT X TO PIAØ
AØ3Ø		CLR	1,X	CLEAR CONTROL REGISTER A ON PIAØ(U8)
AØ32		CLR	3,X	CLEAR CONTROL REGISTER B
AØ34		CLR	, X	A SIDE IS INPUT
AØ36		LDD	#\$FF34	
AØ39		STA	2,X 1,X	B SIDE IS OUTPUT
AØ3B		STB	1,X	ENABLE PERIPHERAL REGISTERS
AØ3D		STB	3,X	AND CA2, CB2 AS OUTPUTS
	*****	****	*****	
AØ3F		LDX	#PIA1	POINT X TO PIA1
AØ42		CLR	1,X	* CLEAR CONTROL REGISTER A ON PIA1(U4)
AØ44		CLR		* CLEAR CONTROL REGISTER B
AØ46		DECA		A - REG NOW HAS \$FE
AØ47		STA	, X	= BITS 1-7 ARE OUTPUTS, BIT Ø IS INPUT
	*			= ON SIDE A
AØ49		LDA	#\$F8	*
AØ4B		STA	2,X	* BITS Ø-2 ARE INPUTS, BITS 3-7 ARE
	*			* OUTPUTS ON B SIDE
AØ4D		STB	1,X	ENABLE PERIPHERAL REGISTERS
AØ4F		STB	3,X	AND CA2, CB2 AS OUTPUTS
AØ51		CLR	3,X 2,X	ZEROS TO 6847
AØ53		LDA	#2	*
AØ55		STA	, X	* MAKE SERIAL OUTPUT MARKING
AØ57		LDA	2,X	READ PORT B OF U4 (TO GET RAM SIZE)
AØ59		LDX	#SAMREG	SAM CONTROL REGISTER ADDR
AØ5C		LDB	#16	16 SAM CONTROL REGISTER BITS
AØ5E	LAØ5E	STA	, X++	ZERO OUT SAM CONTROL REGISTER (CLEAR BITS)
AØ6Ø		DECB	,	DECREMENT REGISTER COUNTER
AØ61		BNE	LAØ5E	BRANCH IF NOT DONE
AØ63		STA	SAMREG+9	SET DISPLAY PAGE AT \$400
AØ66		ANDA		MASK OFF ALL BUT RAM SIZE BIT
AØ68			LAØ6C	BRANCH IF 4K RAM
AØ6A		STA	-5,X	SET FOR 16K DYNAMIC
NUUN		3 1 A	J , A	SEL LOW TOK DIMMITO

AØ6C	LAØ6C	JMP	,U	GO DO A WARM START
AØ6E	BACDST	LDX	#Ø	POINT X TO TOP OF DIRECT PAGE
AØ71	LAØ71	CLR	, χ+	CLEAR FIRST 1K OF RAM
AØ73		CMPX	•	COMPARE TO TOP OF DISPLAY (1K)
AØ76		BNE	LAØ71	BRANCH IF NOT DONE
AØ78		JSR	LA928	CLEAR SCREEN
AØ7B		LDX	#LA1ØD	POINT X TO ROM IMAGE OF DIRECT PAGE VARS
AØ7E		LDU	#CMPMID	POINT U TO RAM DESTINATION
AØ81		LDB	#28	28 BYTES
AØ83		JSR	LA59A	MOVE (B) BYTES FROM (X) TO (U)
AØ86		LDU	#IRQVEC	POINT U TO NON-DIRECT PAGE VARIABLES
AØ89		LDB	#30	30 BYTES
AØ8B		JSR	LA59A	MOVE (B) BYTES FROM (X) TO (U)
AØ8E		LDX	#LB277	ADDR OF SYNTAX ERROR ROUTINE
AØ91		STX	3,U	* SET EXBAS PRIMARY AND SECONDARY
7,051	*	JIX	0,0	* COMMAND INTERPRETATION TABLES TO
AØ93		STX	0 11	* SYNTAX ERROR (U POINTS TO \$12A AT
AUSS	*	311	8,U	
400	*	1.57	UDUE08	* THIS POINT)
AØ95		LDX	#RVECØ	POINT X TO RAM VECTORS
AØ98		LDA	#\$39	OP CODE OF RTS
AØ9A	L4Ø9A	STA	, χ+	PUT RTS'S IN THE RAM VECTORS
AØ9C		CMPX	#RVEC+25*3	
AØ9F		BNE	LAØ9A	NO KEEP INSERTING RTS
AØA1		STA	LINHDR-1	PUT RTS IN \$2D9
AØA4		LDX	VIDRAM+\$200	POINT TO COLOR BASIC'S START OF PROGRAM
AØA7		CLR	, X+	PUT A ZERO AT THE START OF BASIC
AØA9		STX	TXTTAB	BEGINNING OF BASIC PROGRAM
AØAB	LAØAB	LDA	2,X	LOOK FOR END OF PROGRAM
AØAD	LABAD	COMA	<i>L</i> , <i>N</i>	LOOK TOK END OF TROUKAN
			2 V	CTODE IN DAM
AØAE		STA	2,X	STORE IN RAM
AØBØ		CMPA	2,X	IS VALUE IN MEMORY THE SAME AS WHAT WAS
	*			JUST PUT THERE?
AØB2		BNE	LAØBA	IF NOT, THEN IT IS NOT RAM OR THE RAM IS
	*			BAD
AØB4		LEAX	1,X	MOVE TO NEXT RAM LOCATION
AØB6	LAØB6	COM	1,X	RESTORE VALUE OF MEMORY JUST CHANGED
AØB8		BRA	LAØAB	KEEP CHECKING RAM
AØBA	LAØBA	STX	TOPRAM	SET TOP OF RAM POINTER
AØBC		STX	MEMSIZ	TOP OF STRING SPACE
AØBE		STX	STRTAB	START OF STRING VARIABLES
AØCØ			-200,X	* CLEAR 200 BYTES ON A COLD START -
AØC4		STX	FRETOP	* SAVE NEW TOP OF FREE RAM
AØC6		TFR	X,S	PUT STACK THERE (AT MEMEND-200)
AUCO		IIK	۸,3	FUI STACK HIERE (AT HEHEND-200)
+++   ICTINC 0				
*** LISTING 2		D C D	I/ E V T N	CO CHECK KEYDOADD
A1B5		BSR	KEYIN	GO CHECK KEYBOARD
A1B7			LA1B3	LOOP IF NO KEY DOWN
A1B9		LDB	#\$6Ø	BLANK
A1BB			[CURPOS]	BLANK CURRENT CURSOR CHAR ON SCREEN
A1BF	LA1BF	PULS	B,X,PC	
	*			
				OTROVE EDGIA THE MEMBRARD TE A MEM
		ROUTII	NE GETS A KEY	STROKE FROM THE KEYBOARD IF A KEY
	* THIS			STRUKE FROM THE KEYBOARD IF A KEY O TRUE IF THERE WAS NO KEY DOWN.
	* THIS			
A1C1	* THIS * IS DO *	WN. I	Γ RETURNS ZER	O TRUE IF THERE WAS NO KEY DOWN.
A1C1 A1C3	* THIS * IS DO	PSHS	Γ RETURNS ZER Β,Χ	O TRUE IF THERE WAS NO KEY DOWN.  SAVE REGISTERS
A1C1 A1C3 A1C5	* THIS * IS DO *	WN. I	Γ RETURNS ZER	O TRUE IF THERE WAS NO KEY DOWN.

A1C6 A1C8 A1CA A1CD A1CF	LA1C8  ** **	LEAS LDX CLR	B,X,PC -3,S #KEYBUF ,S #\$FE	ALLOCATE 3 STORAGE BYTES ON STACK
A1D1		STB	PIAØ+2	STORE IN COLUMN STROBE REGISTER
A1D4	LA1D4		LA238	GET KEY DATA
A1D6			1,S	TEMP STORE KEY DATA
A1D8		EORA	, X	COMPARE WITH KEY MEMORY DATA
A1DA		ANDA	, X	ACCA=Ø IF THIS KEY WAS DOWN LAST TIME, TOO
A1DC		LDB	1,S	GET NEW KEY DATA
A1DE		STB	, χ+	STORE IT IN KEY MEMORY
A1EØ		TSTA		WAS A NEW KEY DOWN?
A1E1			LA1ED	YES
A1E3		INC	, S	NO, INCREMENT COLUMN COUNTER SET CARRY FLAG
A1E5 A1E6		COMB ROL	PIAØ+2	ROTATE COLUMN STROBE DATA LEFT ONE BIT
A1E9		BCS	LA1D4	ALL COLUMNS CHECKED WHEN ZERO IN THE
ALLS	*	ВОЗ	LAIDT	COLUMN STROBE DATA IS ROTATED INTO THE
	*			CARRY FLAG
A1EB		PULS	B,X,PC	RESTORE REGISTERS
A1ED	LA1ED	LDB	PIAØ+2	GET COLUMN STROBE DATA
				**************************************
	^^ !#!`			
				THE KEY DEPRESSION INTO A NUMBER
<b>Δ1 F</b> Ø		W-50	IN ACCB CORR	ESPONDING TO THE KEY THAT WAS DOWN
A1FØ A1F2		0-50 STB	IN ACCB CORR 2,S	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT
A1FØ A1F2 A1F4		I Ø-5Ø STB LDB	IN ACCB CORR 2,S	ESPONDING TO THE KEY THAT WAS DOWN
A1F2	** FROM	I Ø-5Ø STB LDB	IN ACCB CORR 2,S #\$F8 #8	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8
A1F2 A1F4 A1F6	** FROM	STB LDB ADDB LSRA	IN ACCB CORR 2,S #\$F8 #8	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW
A1F2 A1F4 A1F6 A1F7	** FROM	0-50 STB LDB ADDB LSRA BCC	IN ACCB CORR 2,S #\$F8 #8	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY
A1F2 A1F4 A1F6	** FROM LA1F4 *	M - 50 STB LDB ADDB LSRA BCC ADDB	IN ACCB CORR 2,S #\$F8 #8 LA1F4	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER
A1F2 A1F4 A1F6 A1F7	** FROM LA1F4 *	M-50 STB LDB ADDB LSRA BCC ADDB	IN ACCB CORR 2,S #\$F8 #8 LA1F4 ,S	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9	** FROM LA1F4  *  ****** ** NOW	STB LDB ADDB LSRA BCC ADDB ******	IN ACCB CORR 2,S #\$F8 #8 LA1F4 ,S ***********************************	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9	** FROM LA1F4  *  ****** ** NOW	M 5-50 STB LDB ADDB LSRA BCC ADDB **********************************	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ***********************************	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9	** FROM LA1F4  *  ****** ** NOW	M 0-50 STB LDB ADDB LSRA BCC ADDB **********************************	IN ACCB CORR 2,S #\$F8 #8 LA1F4 ,S ***********************************	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9	** FROM LA1F4  *  ****** ** NOW	M 5-50 STB LDB ADDB LSRA BCC ADDB **********************************	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ***********************************	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF	** FROM LA1F4  *  ****** ** NOW	M 0-50 STB LDB ADDB LSRA BCC ADDB **********************************	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ***************** RT THE VALUE LA245 #26 LA247	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1	** FROM LA1F4  *  ****** ** NOW	M 0-50 STB LDB ADDB LSRA BCC ADDB ****** CONVE BEQ CMPB BHI ORB	IN ACCB CORR 2,S #\$F8 #8 LA1F4 ,S ***********************************	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7	** FROM LA1F4  *  ****** ** NOW	M - 50 STB LDB ADDB LSRA BCC ADDB **********************************	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ********* RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø5 A2Ø7	** FROM LA1F4  *  ****** ** NOW	M - 50 STB LDB ADDB LSRA BCC ADDB ****** CONVEI BEQ CMPB BHI ORB BSR BEQ LDA BNE	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ********* RT THE VALUE LA245 #26 LA247 #\$40 LA247 #\$40 LA22D LA20E CASFLG LA20E	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø5 A2Ø7 A2ØA A2ØC	** FROM LA1F4  *  ****** ** NOW	M 0-50 STB LDB ADDB LSRA BCC ADDB ****** CONVEI BEQ CMPB BHI ORB BSR BEQ LDA BNE ORB	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ********* RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E #\$20	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7 A2ØA A2ØC A2ØE	** FROM LA1F4  *  ****** ** NOW	M 0-50 STB LDB ADDB LSRA BCC ADDB ***********************************	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ********** RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E #\$20 ,S	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7 A2ØA A2ØC A2ØE A2ØE	** FROM LA1F4  *  ****** ** NOW	M - 50 STB LDB ADDB LSRA BCC ADDB ******* CONVEI BEQ CMPB BHI ORB BSR BEQ LDA BNE ORB STB LDX	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S *********** RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E #\$20 ,S DEBVAL	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7 A2ØA A2ØC A2ØE A21Ø A213	** FROM LA1F4  *  ****** ** NOW	M - 50 STB LDB ADDB LSRA BCC ADDB **********************************	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S *********** RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E #\$20 ,S DEBVAL LA7D3	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7 A2ØA A2ØC A2ØE A21Ø A213 A216	** FROM LA1F4  *  ****** ** NOW	M - 50 STB LDB ADDB LSRA BCC ADDB ******* CONVEI BEQ CMPB BHI ORB BSR BEQ LDA BNE ORB STB LDX JSR LDB	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S *********** RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E #\$20 ,S DEBVAL	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7 A2ØA A2ØC A2ØE A21Ø A213	** FROM LA1F4  *  ****** ** NOW	M - 50 STB LDB ADDB LSRA BCC ADDB **********************************	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S *********** RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E CASFLG LA20E #\$20 ,S DEBVAL LA7D3 2,S	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7 A2ØA A2ØC A2ØE A21Ø A213 A216 A218	** FROM LA1F4  *  ****** ** NOW	M 0-50 STB LDB ADDB LSRA BCC ADDB ****** CONVEI BEQ CMPB BHI ORB BSR BEQ LDA BNE ORB STB LDX JSR LDB STB	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ************* RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E CASFLG LA20E #\$20 ,S DEBVAL LA7D3 2,S PIA0+2 LA238	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************
A1F2 A1F4 A1F6 A1F7 A1F9 A1FB A1FD A1FF A2Ø1 A2Ø3 A2Ø5 A2Ø7 A2ØA A2ØC A2ØE A21Ø A213 A216 A218 A218	** FROM LA1F4  *  ****** ** NOW	## OF STB   DB   DB   DB   DB   DB   DB   DB	IN ACCB CORR 2,S #\$F8 #8  LA1F4 ,S ************ RT THE VALUE LA245 #26 LA247 #\$40 LA22D LA20E CASFLG LA20E CASFLG LA20E #\$20 ,S DEBVAL LA7D3 2,S PIA0+2 LA238 1,S	ESPONDING TO THE KEY THAT WAS DOWN TEMP STORE IT TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW GO ON UNTIL A ZERO APPEARS IN THE CARRY ADD IN THE COLUMN NUMBER ************************************

A223 A225 A227 A22A A22B	LA22A LA22B	BNE COM CLRA	#\$12 LA22B CASFLG X,PC	IS SHIFT ZERO DOWN? NO YES, TOGGLE UPPER/LOWER CASE FLAG SET ZERO FLAG TO INDICATE NO NEW KEY DOWN RESTORE REGISTERS
A22D A22F A232 A235 A237	*** TES LA22D	LDA STA LDA	THE SHIFT KE #\$7F PIAØ+2 PIAØ #\$4Ø	Y COLUMN STROBE STORE TO PIA READ KEY DATA CHECK FOR SHIFT KEY, SET ZERO FLAG IF DOWN RETURN
A238	*** REA LA238 **	LDA	KEYBOARD PIAØ	READ PIAØ, PORT A TO SEE IF KEY IS DOWN A BIT WILL BE ZERO IF ONE IS
A23B A23D		ORA TST	#\$8Ø PIAØ+2	MASK OFF THE JOYSTICK COMPARATOR INPUT ARE WE STROBING COLUMN 7?
A24Ø		BMI	LA244	NO
A242	**	ORA	#\$CØ	YES, FORCE ROW 6 TO BE HIGH -THIS WILL CAUSE THE SHIFT KEY TO BE IGNORED
A244	LA244	RTS		RETURN
A245	LA245	LDB	#51	CODE FOR 'AT SIGN'
A247	LA247	LDX		POINT X TO CONTROL CODE TABLE
A24A		CMPB		KEY NUMBER <33?
A24C		BL0	LA264	YES (ARROW KEYS, SPACE BAR, ZERO)
A24E		LDX	#CONTAB-\$54	POINT X TO MIDDLE OF CONTROL TABLE
A251		CMPB	#48	KEY NUMBER > 48?
A253		BHS	LA264	YES (ENTER, CLEAR, BREAK, AT SIGN)
A255		BSR	LA22D	CHECK SHIFT KEY (ACCA WILL CONTAIN STATUS)
A257		CMPB	#43	IS KEY A NUMBER, COLON OR SEMICOLON?
A259		BLS	LA25D	YES
A25B		EORA	#\$4Ø	TOGGLE BIT 6 OF ACCA WHICH CONTAINS THE
	**			SHIFT DATA ONLY FOR SLASH, HYPHEN, PERIOD,
	**			COMMA
A25D	LA25D	TSTA		SHIFT KEY DOWN?
A25E			LA2ØE	YES
A26Ø			#\$10	NO, ADD IN ASCII OFFSET CORRECTION
A262		BKA	LA2ØE	GO CHECK FOR DEBOUNCE
A264	LA264 ** **	ASLB		MULT ACCB BY 2 - THERE ARE 2 ENTRIES IN CONTROL TABLE FOR EACH KEY - ONE SHIFTED, ONE NOT
A265		BSR	LA22D	CHECK SHIFT KEY
A267		BNE	LA26A	NOT DOWN
A269		INCB		ADD ONE TO GET THE SHIFTED VALUE
A26A	LA26A	LDB	В,Х	GET ASCII CODE FROM CONTROL TABLE
A26C		BRA	LA2ØE	GO CHECK DEBOUNCE
*** LISTING 3				
A2C3		BSR	LA2FB	SET OUTPUT TO MARKING
A2C5		ASLA		SEND 7 BITS AND ONE STOP BIT (BIT 7=0)
A2C6		LDB	#8	SEND 8 BITS
A2C8	LA2C8	PSHS	В	SAVE BIT COUNTER
A2CA		CLRB		CLEAR DA IMAGE 1 ZEROS TO DA WHEN SENDING
	*			RS-232 DATA

A2CB	*	LSRA		ROTATE NEXT BIT OF OUTPUT CHARACTER TO CARRY FLAG
A2CC		ROLB		* ROTATE CARRY FLAG INTO BIT ONE
A2CD		ROLB		* AND ALL OTHER BITS SET TO ZERO
A2CE		STB	DA	STORE IT TO DA CONVERTER
A2D1		BSR	LA3Ø2	GO WAIT A WHILE
A2D3		NOP	LAJUZ	do wall a wille
A2D4		NOP		
A2D5		NOP		
A2D6		BSR	LA3Ø2	GO WAIT SOME MORE
		PULS		
A2D8 A2DA		DECB	D	GET BIT COUNTER SENT ALL 8 BITS?
			1 4 2 C O	
A2DB		BNE BSR	LA2C8 LA2FB	NO
A2DD				SEND STOP BIT (ACCB=0)
A2DF			CC,A	RESTORE OUTPUT CHARACTER & INTERRUPT STATS
A2E1			#CR	IS IT A CARRIAGE RETURN?
A2E3		BEQ	LA2ED	YES
A2E5		INC	LPTPOS	INCREMENT CHARACTER COUNTER
A2E7		LDB	LPTPOS	CHECK FOR END OF LINE PRINTER LINE
A2E9			LPTWID	AT END OF LINE PRINTER LINE?
A2EB	1 4 2 5 5	BLO	LA2F3	NO
A2ED	LA2ED		LPTPOS	RESET CHARACTER COUNTER *
A2EF		BSR	LA3Ø5	
A2F1	1 4 0 5 0	BSR	LA3Ø5	* DELAY FOR CARRIAGE RETURN
A2F3	LA2F3	LDB	PIA1+2	WAIT FOR HANDSHAKE
A2F6		LSRB	1 4050	CHECK FOR RS232 STATUS
A2F7		BCS	LA2F3	NOT YET READY
A2F9		PULS	B,X,PC	RESTORE REGISTERS
*** LISTING 4				
B3ED	INTCNV	LDA	FPØEXP	GET FPAØ EXPONENT
B3ED B3EF	INTCNV	CMPA		GET FPAØ EXPONENT * COMPARE TO 32768 - LARGEST INTEGER
B3ED B3EF B3F1	INTCNV			
B3ED B3EF B3F1 B3F3	INTCNV	CMPA	# <b>\$9</b> Ø	* COMPARE TO 32768 - LARGEST INTEGER * EXPONENT AND BRANCH IF FPAØ < 32768 POINT X TO FP VALUE OF -32768
B3ED B3EF B3F1 B3F3 B3F6	INTCNV	CMPA BCS	#\$9Ø LB3FB	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ
B3ED B3EF B3F1 B3F3 B3F6 B3F9	INTCNV	CMPA BCS LDX JSR BNE	#\$90 LB3FB #LB3DF LBC96 LB44A	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =
B3ED B3EF B3F1 B3F3 B3F6	INTCNV LB3FB	CMPA BCS LDX JSR	#\$9Ø LB3FB #LB3DF LBC96	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER
B3ED B3EF B3F1 B3F3 B3F6 B3F9		CMPA BCS LDX JSR BNE	#\$90 LB3FB #LB3DF LBC96 LB44A	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB	LB3FB	CMPA BCS LDX JSR BNE JSR LDD RTS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPA0+3	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ	LB3FB * EVALL	CMPA BCS LDX JSR BNE JSR LDD RTS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPA0+3	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B400	LB3FB	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AI	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B400 B401 B403	LB3FB * EVALL	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AI LDB LDA	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B400 B401 B403 B405	LB3FB * EVALL	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7	LB3FB * EVALU LB4Ø1	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8	LB3FB * EVALL	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4Ø8	LB3FB * EVALU LB4Ø1	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX PSHS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4ØA B4ØA	LB3FB * EVALU LB4Ø1	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX PSHS BSR	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPA0+3 N ARRAY VARIA DIMFLG VALTYP A,B VARNAM B,X LB3E4	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4Ø8	LB3FB * EVALU LB4Ø1	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX PSHS BSR	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4ØA B4ØA	LB3FB * EVALU LB4Ø1 LB4Ø8	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX PSHS BSR	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPA0+3 N ARRAY VARIA DIMFLG VALTYP A,B VARNAM B,X LB3E4	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)  PULL OFF VARIABLE NAME, DIMENSION COUNTER
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B400 B401 B403 B405 B407 B408 B407 B408 B40A B40C B40E	LB3FB * EVALU LB4Ø1 LB4Ø8	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AI LDB LDA PSHS CLRB LDX PSHS BSR PULS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B VARNAM B,X LB3E4 B,X,Y	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)  PULL OFF VARIABLE NAME, DIMENSION COUNTER  ARRAY FLAG
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B400 B401 B403 B405 B407 B408 B407 B408 B40A B40C B40E	LB3FB * EVALU LB4Ø1 LB4Ø8	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AI LDB LDA PSHS CLRB LDX PSHS BSR PULS STX LDU	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3  N ARRAY VARIA DIMFLG VALTYP A,B  VARNAM B,X LB3E4 B,X,Y  VARNAM FPAØ+2	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)  PULL OFF VARIABLE NAME, DIMENSION COUNTER  ARRAY FLAG  SAVE VARIABLE NAME AND VARIABLE TYPE  GET DIMENSION LENGTH
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4ØA B4ØC B4ØE B41Ø B412 B414	LB3FB * EVALU LB4Ø1 LB4Ø8	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AI LDB LDA PSHS CLRB LDX PSHS BSR PULS STX LDU PSHS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3  N ARRAY VARIA DIMFLG VALTYP A,B  VARNAM B,X LB3E4 B,X,Y  VARNAM FPAØ+2	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)  PULL OFF VARIABLE NAME, DIMENSION COUNTER  ARRAY FLAG  SAVE VARIABLE NAME AND VARIABLE TYPE  GET DIMENSION LENGTH  SAVE DIMENSION LENGTH, ARRAY FLAG,  VARIABLE TYPE
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4Ø7 B4Ø8 B4ØA B4ØC B4ØE B41Ø B412 B414	LB3FB * EVALU LB4Ø1 LB4Ø8	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX PSHS BSR PULS STX LDU PSHS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B VARNAM B,X LB3E4 B,X,Y VARNAM FPAØ+2 Y,U	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)  PULL OFF VARIABLE NAME, DIMENSION COUNTER  ARRAY FLAG  SAVE VARIABLE NAME AND VARIABLE TYPE  GET DIMENSION LENGTH, ARRAY FLAG,  VARIABLE TYPE  INCREASE DIMENSION COUNTER
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4Ø7 B4Ø8 B4ØA B4ØC B4ØE B41Ø B412 B414	LB3FB * EVALU LB4Ø1 LB4Ø8	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX PSHS BSR PULS STX LDU PSHS INCB JSR	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B VARNAM B,X LB3E4 B,X,Y VARNAM FPAØ+2 Y,U GETCCH	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)  PULL OFF VARIABLE NAME, DIMENSION COUNTER  ARRAY FLAG  SAVE VARIABLE NAME AND VARIABLE TYPE  GET DIMENSION LENGTH  SAVE DIMENSION LENGTH, ARRAY FLAG,  VARIABLE TYPE  INCREASE DIMENSION COUNTER  GET CURRENT INPUT CHARACTER
B3ED B3EF B3F1 B3F3 B3F6 B3F9 B3FB F3FE B4ØØ B4Ø1 B4Ø3 B4Ø5 B4Ø7 B4Ø8 B4Ø7 B4Ø8 B4ØA B4ØC B4ØE B41Ø B412 B414	LB3FB * EVALU LB4Ø1 LB4Ø8	CMPA BCS LDX JSR BNE JSR LDD RTS JATE AN LDB LDA PSHS CLRB LDX PSHS BSR PULS STX LDU PSHS	#\$90 LB3FB #LB3DF LBC96 LB44A LBCC8 FPAØ+3 N ARRAY VARIA DIMFLG VALTYP A,B VARNAM B,X LB3E4 B,X,Y VARNAM FPAØ+2 Y,U GETCCH	* COMPARE TO 32768 - LARGEST INTEGER  * EXPONENT AND BRANCH IF FPAØ < 32768  POINT X TO FP VALUE OF -32768  COMPARE -32768 TO FPAØ  'FC' ERROR IF NOT =  CONVERT FPAØ TO A TWO BYTE INTEGER  GET THE INTEGER  BLE  GET ARRAY FLAG  GET VARIABLE TYPE  SAVE THEM ON THE STACK  RESET DIMENSION COUNTER  GET VARIABLE NAME  SAVE VARIABLE NAME  SAVE VARIABLE NAME AND DIMENSION COUNTER  EVALUATE EXPRESSION (DIMENSION LENGTH)  PULL OFF VARIABLE NAME, DIMENSION COUNTER  ARRAY FLAG  SAVE VARIABLE NAME AND VARIABLE TYPE  GET DIMENSION LENGTH, ARRAY FLAG,  VARIABLE TYPE  INCREASE DIMENSION COUNTER

COLOR BASIC UNRAVELLED II	APPENDIX I	ORIGIN:SPECTRAL ASSOC
	COLOR BASIC 1.0 DIFFERENCES	REVISED:12/13/1999 WALTER K ZYDHEK

B41D	STB	TMPLOC	SAVE DIMENSION COUNTER
B41F	JSR	LB267	SYNTAX CHECK FOR A ")"
B422	PULS	A,B	* RESTORE VARIABLE TYPE AND ARRAY
B424	STA	VALTYP	* FLAG - LEAVE DIMENSION LENGTH ON STACK
R426	STB	DIMFLG	*

Listed below are the differences between Color BASIC Version 1.1 and Version 1.2  $\,$ 

CHANGE	ADDRESS	Version 1.0	Version 1.2
Α	\$AØØ1	\$C1	\$CB
В	\$A114	\$57	\$58
С	\$A155	\$31	\$32
D	\$A15E	\$30	\$32
E	\$A1B5-\$A26D	SEE LISTING 5	
F	\$A2C3-\$A2FA	SEE LISTING 6	
G	\$A56A	\$C1	\$CB
Н	\$ADFD	\$C1	\$CB
J	\$B23F	\$E8	\$E3
K	\$B38E	\$72	\$75
L	\$B3ED-\$B427	SEE LISTING 7	
М	\$B9D6	\$2B BMI	\$25 BCS

Change A is a branch length change caused by the keyboard driver mod.

Change B modified the line printer baud rate.

Change C is the version number.

Change D is the copyright year.

Change E is a major change (Listing 5) to the keyboard driver allowing a quick scan of the keyboard if no keys are down.

Change F is also a major change to the line printer driver which will not allow any transmission until the receiving device is ready (Listing 6).

Change G speeds up the INKEY\$ command by not entering the keyboard scan routine if no keys are depressed.

Change H speeds up the BREAK key check routine by not entering the keyboard scan routine if no keys are depressed.

Change J is a minor change which slightly speeds up the expression evaluation routine.

Change K is merely a different length branch caused by change L.

Change L (Listing 7) causes a numeric variable type check to be done before the INTCNV (FPA $\emptyset$  to integer) routine.

Change M fixed a minor bug in the floating point addition routine.

*** LISTING 5				
A1B5		BSR	KEYIN	GO CHECK KEYBOARD
A1B7		BEQ	LA1B3	LOOP IF NO KEY DOWN
A1B9		LDB	#\$60	BLANK
A1BB		STB	[CURPOS]	BLANK CURRENT CURSOR CHAR ON SCREEN
A1BF	LA1BF	PULS	B,X,PC	

A21Ø

A213

A215

A217

A218

* THIS ROUTINE GETS A KEYSTROKE FROM THE KEYBOARD IF A KEY * IS DOWN. IT RETURNS ZERO TRUE IF THERE WAS NO KEY DOWN. A1C1 KEYIN PSHS B,X,U SAVE REGISTERS A1C3 BSR LA1C8 **GET KEYSTROKE** A1C5 TSTA SET FLAGS A1C6 PULS B, X, U, PC RESTORE REGISTERS LA1C8 LDU #PIAØ POINT TO PIAØ A1C8 A1CA LDX #KEYBUF KEYBOARD MEMORY BUFFER A1CE CLRA * CLEAR CARRY FLAG, SET COLUMN COUNTER A1CF DECA * (ACCA) TO \$FF A1DØ PSHS X,A SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK A1D2 STA 2,U INITIALIZE COLUMN STROBE TO \$FF A1D4 FCB SKP1 SKIP ONE BYTE A1D5 LA1D5 COMB SET CARRY FLAG A1D6 R0L 2,U * ROTATE COLUMN STROBE DATA LEFT 1 BIT, A1D8 BCC LA1BF * CARRY INTO BIT Ø-RETURN IF 8 BITS DONE A1DA INC ,S INCREMENT COLUMN POINTER A1DC BSR LA239 READ KEYBOARD DATA ROW A1DE STA 1,S TEMP STORE KEY DATA A1EØ EORA ,X SET ANY BIT WHERE A KEY HAS MOVED A1E2 ANDA **,** X ACCA=Ø IF NO NEW KEY DOWN, <7Ø IF KEY WAS RELEASED A1E4 LDB 1.S GET NEW KEY DATA , χ+ A1E6 STB STORE IT IN KEY MEMORY WAS A NEW KEY DOWN? A1F8 **TSTA** BEO NO-CHECK ANOTHER COLUMN A1E9 LA1D5 * GET COLUMN STROBE DATA AND A1EB LDB 2,U * TEMP STORE IT ON THE STACK A1ED STB 2,S ******************* ** THIS ROUTINE CONVERTS THE KEY DEPRESSION INTO A NUMBER ** FROM Ø-5Ø IN ACCB CORRESPONDING TO THE KEY THAT WAS DOWN A1EF LDB #\$F8 TO MAKE SURE ACCB=Ø AFTER FIRST ADDB #8 ADDB #8 ADD 8 FOR EACH ROW OF KEYBOARD A1F1 LA1F1 A1F3 LSRA ACCA CONTAINS THE ROW NUMBER OF THIS KEY ADD 8 FOR EACH ROW A1D4 BCC LA1F1 GO ON UNTIL A ZERO APPEARS IN THE CARRY ADDB ,S ADD IN THE COLUMN NUMBER A1F6 ******************* ** NOW CONVERT THE VALUE IN ACCB INTO ASCII THE 'AT SIGN' KEY WAS DOWN A1F8 BE0 LA244 A1FA CMPB #26 WAS IT A LETTER? A1FC BHI LA246 NO #\$4Ø YES, CONVERT TO UPPER CASE ASCII A1FE ORB LA22E CHECK FOR THE SHIFT KEY A2ØØ BSR A2Ø2 BEQ LA2ØB IT WAS DOWN A2Ø4 I DA CASFLG NOT DOWN. CHECK THE UPPER/LOWER CASE FLAG A2Ø7 BNE LA2ØB UPPER CASE A2Ø9 ORB #\$2Ø CONVERT TO LOWER CASE A2ØB LA2ØB STB ,S TEMP STORE ASCII VALUE A2ØD LDX DEBVAL GET KEYBOARD DEBOUNCE

GO WAIT A WHILE

* SET COLUMN STROBE TO ALL ONES (NO

= INCR ROW DATA, ACCA NOW Ø IF NO JOYSTK

= BUTTON DOWN. BRANCH IF JOYSTK BUTTON DN

* STROBE) AND READ KEYBOARD

JSR

LDB

BSR

INCA

BNE

LA7D3

#\$FF

LA237

LA22Ø

A21A A21C A21E A22Ø	LA21A LA22Ø	BSR CMPA	2,S LA237 1,S	GET COLUMN STROBE DATA READ A KEY IS IT THE SAME KEY AS BEFORE DEBOUNCE? PUT THE ASCII VALUE OF KEY BACK IN ACCA
A222 A224 A226	LAZZW	BNE CMPA BNE	LA22B #\$12 LA22C	NOT THE SAME KEY IS SHIFT ZERO DOWN? NO
A228 A22B A22C	LA22B LA22C	CLRA	CASFLG X,PC	YES, TOGGLE UPPER/LOWER CASE FLAG SET ZERO FLAG TO INDICATE NO NEW KEY DOWN REMOVE TEMP STORAGE SLOTS FROM STACK
	*** TES	T FOR	THE SHIFT KE	
A22E	LA22E	LDA		COLUMN STROBE
A23Ø		STA	2,0	STORE TO PIA
A232		LDA	, U	READ KEY DATA CHECK FOR SHIFT KEY, SET ZERO FLAG IF DOWN
A234 A236		RTS	#\$40	RETURN
A230		KIS		RETURN
	*** REA	D THE	KEYBOARD	
A237	LA237	STB	2,U	SAVE NEW COLUMN STROBE VALUE
A239	LA239 **		•	READ PIAØ, PORT A TO SEE IF KEY IS DOWN A BIT WILL BE ZERO IF ONE IS
A23B		ORA	#\$80	MASK OFF THE JOYSTICK COMPARATOR INPUT
A23D			2,U	ARE WE STROBING COLUMN 7?
A23F A241		BMI	LA243	NO
A241	**	ORA	#\$CØ	YES, FORCE ROW 6 TO BE HIGH -THIS WILL CAUSE THE SHIFT KEY TO BE IGNORED
A243	LA243	RTS		RETURN
A244	LA244	LDB	#51	CODE FOR 'AT SIGN'
A246	LA246	LDX		POINT X TO CONTROL CODE TABLE
A249			#33	KEY NUMBER <33?
A24B		BLO		YES (ARROW KEYS, SPACE BAR, ZERO)
A24D		LDX		POINT X TO MIDDLE OF CONTROL TABLE
A25Ø			#48	KEY NUMBER > 48?
A252				YES (ENTER, CLEAR, BREAK, AT SIGN)
A254				CHECK SHIFT KEY (ACCA WILL CONTAIN STATUS)
A256 A258			#43 LA25C	IS KEY A NUMBER, COLON OR SEMICOLON? YES
A25A			#\$4Ø	TOGGLE BIT 6 OF ACCA WHICH CONTAINS THE
KLJK	**	LOKA	// <del>4 + 5</del>	SHIFT DATA ONLY FOR SLASH, HYPHEN, PERIOD,
	**			COMMA
A25C	LA25C	TSTA		SHIFT KEY DOWN?
A25D		BEQ	LA2ØB	YES
A25F		ADDB	#\$1Ø	NO, ADD IN ASCII OFFSET CORRECTION
A261		BRA	LA2ØB	GO CHECK FOR DEBOUNCE
A263	LA263 ** **	ASLB		MULT ACCB BY 2 - THERE ARE 2 ENTRIES IN CONTROL TABLE FOR EACH KEY - ONE SHIFTED, ONE NOT
A264		BSR	LA22E	CHECK SHIFT KEY
A266		BNE	LA269	NOT DOWN
A268		INCB		ADD ONE TO GET THE SHIFTED VALUE
A269	LA269	LDB	B,X	GET ASCII CODE FROM CONTROL TABLE
A26B		BRA	LA2ØB	GO CHECK DEBOUNCE
A26D		FCB	Ø	WASTED SPACE IN VERSION 1.1

*** LISTING 6				
A2C3		BSR	LA2FB	SET OUTPUT TO MARKING
A2C5		CLRB		*
A2C6		BSR	LA2FD	* TRANSMIT ONE START BIT
A2C8		LDB	#8	SEND 8 BITS
A2CA	LA2CA	PSHS	В	SAVE BIT COUNTER
A2CC		CLRB		CLEAR DA IMAGE 1 ZEROS TO DA WHEN SENDING
	*			RS-232 DATA
A2CD		LSRA		ROTATE NEXT BIT OF OUTPUT CHARACTER TO
	*			CARRY FLAG
A2CE		ROLB		* ROTATE CARRY FLAG INTO BIT ONE
A2CF		ASLB		* AND ALL OTHER BITS SET TO ZERO
A2DØ		BSR	LA2FD	TRANSMIT DATA BYTE
A2D2		PULS		GET BIT COUNTER
A2D4		DECB		SENT ALL 8 BITS?
A2D5		BNE	LA2CA	NO
A2D7		BSR	LA2FB	SEND STOP BIT (ACCB=0)
A2D9			CC,A	RESTORE OUTPUT CHARACTER & INTERRUPT STATS
A2DB			#CR	IS IT A CARRIAGE RETURN?
A2DD		BEQ	LA2E7	YES
A2DF		INC	LPTPOS	INCREMENT CHARACTER COUNTER
A2E1		LDB	LPTPOS	CHECK FOR END OF LINE PRINTER LINE
A2E3		CMPB	LPTWID	AT END OF LINE PRINTER LINE?
A2E5		BLO	LA2ED	NO
A2E7	LA2E7	CLR	LPTPOS	RESET CHARACTER COUNTER
A2E9		BSR	LA3Ø5	*
A2EB		BSR	LA3Ø5	* DELAY FOR CARRIAGE RETURN
A2ED	LA2ED	LDB	PIA1+2	WAIT FOR HANDSHAKE
A2FØ		LSRB		CHECK FOR RS232 STATUS
A2F1		BCS	LA2ED	NOT YET READY
A2F3			B,X,PC	RESTORE REGISTERS
A2F5		FDB	ø,ø,ø	WASTED SPACE IN VERSION 1.1
*** LISTING 7				
B3ED	INTCNV	LDA	FPØEXP	
B3EF		CMPA	#\$9Ø	* COMPARE TO 32768 - LARGEST INTEGER
B3F1		BCS	LB3FB	* EXPONENT AND BRANCH IF FPAØ < 32768
B3F3		LDX	#LB3DF	POINT X TO FP VALUE OF -32768
B3F6		JSR		COMPARE -32768 TO FPAØ
B3F9		BNE	LB44A	'FC' ERROR IF NOT =
B3FB	LB3FB	JSR	LBCC8	CONVERT FPAØ TO A TWO BYTE INTEGER
F3FE		LDD	FPAØ+2	GET THE INTEGER
B400		RTS		
			N ARRAY VARIA	
B4Ø1	LB4Ø1	LDB	DIMFLG	GET ARRAY FLAG
B4Ø3		LDA	VALTYP	GET VARIABLE TYPE
B4Ø5		PSHS	A,B	SAVE THEM ON THE STACK
B4Ø7		CLRB		RESET DIMENSION COUNTER
B4Ø8	LB4Ø8	LDX	VARNAM	GET VARIABLE NAME
B4ØA		PSHS		SAVE VARIABLE NAME AND DIMENSION COUNTER
B4ØC		BSR	LB3E4	EVALUATE EXPRESSION (DIMENSION LENGTH)
B4ØE		PULS	B,X,Y	PULL OFF VARIABLE NAME, DIMENSION COUNTER
	*			ARRAY FLAG
B410	*	STX	VARNAM	SAVE VARIABLE NAME AND VARIABLE TYPE
B410 B412	*	LDU	FPAØ+2	SAVE VARIABLE NAME AND VARIABLE TYPE GET DIMENSION LENGTH
B410	*		FPAØ+2	SAVE VARIABLE NAME AND VARIABLE TYPE

COLOR BASIC UNRAVELLED II	APPENDIX J	ORIGIN:SPECTRAL ASSOC
	COLOR BASIC 1.1 DIFFERENCES	REVISED:12/13/1999 WALTER K ZYDHEK

B416 B417	INCB JSR GETCCH	INCREASE DIMENSION COUNTER GET CURRENT INPUT CHARACTER
B419	CMPA #',	CHECK FOR ANOTHER DIMENSION
B41B	BEQ LB4Ø8	BRANCH IF MORE
B41D	STB TMPLOC	SAVE DIMENSION COUNTER
B41F	JSR LB267	SYNTAX CHECK FOR A ")"
B422	PULS A,B	* RESTORE VARIABLE TYPE AND ARRAY
B424	STA VALTYP	* FLAG - LEAVE DIMENSION LENGTH ON STACK
B426	STB DIMFLG	*

# DISPLAY CHARACTER SET

HEX VALUE			HEX VALUE			HEX VALUE			
Non-		CHARACTER	Non-		CHARACTER	Non-		CHARACTER	
Inverted	Inverted		Inverted	Inverted		Inverted	Inverted		
ØØ	40	@	18	58	Х	3Ø	40	Ø	
Ø1	41	Α	19	59	Υ	31	41	1	
Ø2	42	В	1A	5 A	Z	32	42	2 3	
Ø3	43	С	1B	5B	[	33	43		
Ø4	44	D	1C	5 C	\	34	44	4	
Ø5	45	Е	1 D	5 D	]	35	45	5	
Ø6	46	F	1 E	5 E	<b>^</b>	36	46	6	
Ø7	47	G	1 F	5 F	<b>←</b>	37	47	7	
Ø8	48	Н	20	60		38	48	8	
Ø9	49	I	21	61	!	39	49	9	
ØA	4A	J	22	62	"	3A	4 A	:	
ØB	4B	K	23	63	#	3B	4B	;	
ØC	4 C	L	24	64	\$	3C	4 C	<	
ØD	4 D	М	25	65	%	3 D	4 D	=	
ØE	4 E	N	26	66	&	3E	4 E	>	
ØF	4 F	0	27	67	•	3F	4 F	?	
10	50	Р	28	68	(				
11	51	Q	29	69	)				
12	52	R	2 A	6 A	*				
13	53	S	2B	6B	+				
14	54	T	2C	6C	,				
15	55	U	2 D	6 D	-				
16	56	V	2 E	6 E	•				
17	57	W	2 F	6 F	/				